Google Cloud

DORA
DEVOPS RESEARCH & ASSESSMENT

2022 ACCELERATE

State of DevOps Report

スポンサー

armory

BROADCOM SOFTWARE

circleci

Deloitte.

GitLab

JET BRAINS

JFrog

Liquibase

Octopus Deploy

sysdig

目次

01

エグゼクティブ サマリー

Derek DeBellis

Claire Peters

過去 8 年間、私たちは Accelerate State of DevOps レポートを作成し、その過程で 33,000 人のプロフェッショナルから意見を聞いてきました。私たちの研究は、能力と実践によって、私たちが DevOps にとって重要と考える成果をどのように予測できるかを検証することに重点を置いています。

- ソフトウェア デリバリーのパフォーマンス - ソフトウェア デリバリーのパフォーマンスに関する 4 つの主要指標: デプロイ頻度、変更のリードタイム、変更時の障害率、サービス復旧時間。

- 運用パフォーマンス - 5 つ目の重要指標である信頼性。

- 組織パフォーマンス - 組織がパフォーマンスと収益性の目標をどの程度達成しているか。

私たちは、燃え尽き症候群や、従業員が自分のチームを推薦する傾向など、他の成果の根底にある要因にも着目しています。

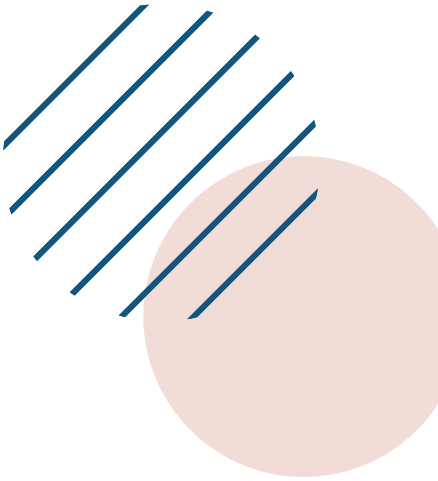
ソフトウェア サプライ チェーンの保護

2021 年、私たちは、多くの重要な成果を達成するため、ソフトウェア サプライ チェーンの保護が重要であると認識しました。

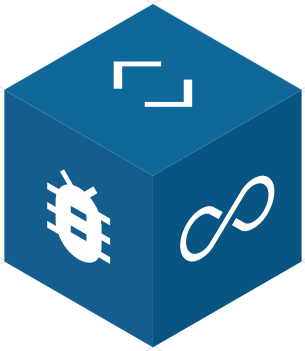
今年は、ソフトウェア サプライ チェーンのセキュリティについてより深く調査しており、これを調査と報告の主要テーマに据えています。ソフトウェア サプライ チェーンのセキュリティの開発に役立つ技術的取り組みを探るため、Supply Chain Levels for Secure Artifacts (SLSA) フレームワークを活用しました。また、ソフトウェア サプライ チェーンの保護に関連する姿勢、プロセス、非技術的取り組みを探るため、米国国立標準技術研究所の Secure Software Development Framework (NIST SSDF) も活用しました。

その結果、組織のアプリケーション開発におけるセキュリティ対策の最大の予測因子は、技術的なものではなく文化的なものであることがわかりました。パフォーマンスを重視する、信頼性が高く、非難されることが少ない文化は、権力や規則を重視する、信頼性が低く、非難されることが多い文化に比べて、新しいセキュリティ対策が平均以上に採用される傾向が 1.6 倍も強くなっていました。また、デプロイ前のセキュリティ スキャンが脆弱な依存関係を検出するのに有効であり、その結果、本番コードの脆弱性が減少したことを示唆する早期の兆候も確認しました。

優れたアプリケーション開発時セキュリティ対策の採用には、これ以外の利点もありました。このようなセキュリティ対策の確立に重点を置いているチームでは、開発者の燃え尽き症候群が減少することがわかりました。セキュリティ対策の水準が低いチームは、セキュリティ対策の水準が高いチームに比べて、燃え尽き症候群の程度が大きくなる確率が 1.4 倍高くなることがわかりました。[1] セキュリティ対策の確立に重点を置いているチームは、自分のチームを他人に推薦する傾向が著しく強いことがわかっています。さらに、SLSA に関連するセキュリティ対策により、組織の業績とソフトウェア デリバリーのパフォーマンスの両方を積極的に予測できますが、この効果が完全に現れるには、強力な継続的インテグレーション機能を導入する必要があります。



[1] 私たちは、スコア (セキュリティなど) の標準偏差が 1 以上の場合にこの統計データにおいて水準が高く、スコアの標準偏差が -1 以下の場合に水準が低いと解釈しています。



組織パフォーマンスの推進要素

上記のセキュリティ対策以外に、組織パフォーマンスに影響を与える重要要素は、以下の範疇に収まる傾向にあります。

組織やチームにおける文化

Westrum が定義するところの信頼性が高く、非難されることが少ない文化圏は、より高い組織パフォーマンスを発揮する傾向にあります。同様に、チームが資金面とリーダーシップの面で支援を受けられていると感じている組織は、より高い組織パフォーマンスを発揮する傾向にあります。チームの安定度と、そのチームに関する肯定的な認識（そのチームを推奨する傾向）も、より高度な組織パフォーマンスにつながる傾向にあります。最後に、柔軟な業務形態を認めている企業も、より高い組織パフォーマンスを発揮する傾向にあります。

信頼性

私たちが信頼性エンジニアリングと関連付ける手法（明確な信頼性目標、顕著な信頼性指標など）と、信頼性に関する期待に応えられていると人々が報告する程度は、ともにより高度な組織パフォーマンスにつながる有力な要素です。

クラウド

私たちは、クラウドの使用状況が組織パフォーマンスの予測因子となることを確認しました。最初からソフトウェアをクラウド上で、クラウド向けに構築している企業は、より高い組織パフォーマンスを発揮する傾向にあります。プライベート クラウド、パブリック クラウド、ハイブリッド クラウド、または異なるクラウドの複合サービスを使用すると、オンプレミス サーバーのみを使用する場合より組織パフォーマンスが高くなります。複数のパブリック クラウドを使用する組織は、そうではない組織より平均を上回る組織パフォーマンスを発揮する確率が 1.4 倍高くなります。

クラウドの使用状況は、私たちのデータセットの他の要因を通じて組織パフォーマンスに影響を与えるようにも思われます。一つの例として、サプライ チェーン セキュリティが挙げられます。これについては、おそらくはクラウド プロバイダがビルドとデプロイの自動化などの多くの SLSA 手法の構成要素の利用を推奨し、これを提供していることから、パブリック クラウドを使用している組織の方が SLSA 手法を導入する傾向も高いことがわかりました。[2,3] さらに、クラウド プラットフォームを使用することで、最終的に組織パフォーマンス向上につながる多くの機能と手法をチームが受け継げるようになります。

[2] Jung, Sun Jae.「Introduction to Mediation Analysis and Examples of Its Application to Real-world Data（英語）」Journal of preventive medicine and public health = Yebang Uihakhoe chi vol. 54、3（2021 年）: 166〜172. doi:10.3961/jpmph.21.069

[3] Carrión、Gabriel Cepeda、Christian Nitzl、および José L. Roldán.「Mediation analyses in partial least squares structural equation modeling: Guidelines and empirical examples（英語）」Partial least squares path modeling.Springer, Cham、2017 年。173〜195。

状況が重要

長い間、DORA は、効果がより幅広いチームの状況に左右されることを考慮してきました。私たちは、チームの特性（プロセス、強み、制約、目標）と、業務環境を理解することが重要であると考えています。たとえば、ある状況では役に立つ技術的能力が、別の状況では不利になることがあります。今年は、このような仮説に基づいた条件をインタラクションという形で明示的にモデル化することに注力しました。このような仮説の多くが今年のデータで裏付けられています。

- 高いソフトウェア デリバリー パフォーマンスは、運用パフォーマンスも高い場合のみ、組織パフォーマンスに良い影響を与えます。ユーザーが持つ信頼性の期待度にサービスが届かない場合は、迅速なデリバリーが意味を成さないことがあります。

- SLSA フレームワークが推奨するものなどのソフトウェア サプライ チェーン セキュリティ対策を導入することで、継続的インテグレーションがしっかりと確立されている場合、ソフトウェア デリバリー パフォーマンスに好影響がもたらされます。継続的インテグレーション機能が実装されていない場合、ソフトウェア デリバリー パフォーマンスとセキュリティ対策が競合することがあります。

- チームが信頼性目標を達成する能力に対するサイト信頼性エンジニアリング（SRE）対策の影響は、非線形です。SRE 対策を導入しても、チームが特定レベルの SRE 成熟度に達するまでは、信頼性に好影響がもたらされません。チームが特定レベルの SRE 成熟度に達するまでは、SRE と信頼性目標達成の関係を確認できません。しかしながら、チームの SRE 採用が増えるにつれ、チームが SRE の使用により信頼性を高い確度で予測でき始める転換点に到達します。信頼性が向上すると、組織パフォーマンスにも変化が出てきます。

- 技術的能力には相乗効果があります。継続的デリバリーとバージョン管理は、高度なソフトウェア デリバリー パフォーマンスを発揮する互いの能力を高めあいます。継続的デリバリー、疎結合アーキテクチャ、バージョン管理、継続的インテグレーションを組み合わせることで、これらを単純に合計した場合よりも優れたソフトウェア デリバリー パフォーマンスが発揮されます。

デリバリーが依存する条件と、チームのより幅広い状況を理解する必要性から、以下の 2021 年からの見解と類似した結論が導き出されます。

「意味のある改善を行うには、チームが継続的な改善という理念を採用する必要があります。ベンチマークを使用して現在の状況を測定し、調査された能力に基づいて制約を識別し、それらの制約を緩和するための改善を試みてください。探求には勝利と失敗の両方が伴いますが、どちらの場合もチームは得られた教訓の結果として有意義な行動をとることができます。」

実際、私たちは今年、この基本理念と深く一致する効果を確認しました。それは、継続的な改善の必要性を認識しているチームは、そうでないチームよりも高い組織パフォーマンスを発揮する傾向がある、ということです。

つまり、チームは適応を継続したうえで、ソフトウェア開発手法を試す必要があります。

これを実行するチームは、総じてより高い組織パフォーマンスを発揮するからです。常にというわけではありませんが（ある組織でうまくいくことが、別の組織でうまくいくとは限りません）、ほとんどの場合はそうです。DevOps 手法を用いた独自の試行錯誤に取り組む際には、自分のチームに合った手法を磨き上げる中で、時折失敗することも念頭に置いておきましょう。

今年、私たちはデータの中にいくつかの予想外の点も発見しましたが、それが何であるかは後ほど説明します。

> 継続的な改善の必要性を認識しているチームは、そうでないチームよりも高い組織パフォーマンスを発揮する傾向があります。

02

他社との比較

Derek DeBellis

自分のチームが業界の他のチームと比較するとどの程度だろうかとお考えでしょうか？このセクションでは、DevOps のパフォーマンスに関する最新のベンチマーク評価について説明します。私たちは、チームがどのようにソフトウェア システムを開発、提供、運用しているかを調査し、DevOps パフォーマンスの最も一般的な組み合わせを反映したクラスタに回答者を分類しています。

今年は、2 種類のクラスタ分類手法を採用しています。1 つ目は、過去の事例に基づくものです。このクラスタ分類手法は、ソフトウェア デリバリー パフォーマンスを把握する 4 つの指標（リードタイム、デプロイ頻度、サービス復旧時間、変更時の障害率）に基づくクラスタ作成に重点を置いています。以下に、これらのそれぞれについて概説します。この手法の目的は、チームの現在のパフォーマンスを定量化し、他のチームと比較できるようにすることです。

2 つ目のクラスタ分類手法では、5 つ目の指標として、運用パフォーマンスを把握するための「信頼性」を採用しています。なぜ、クラスタ分析に新しい指標を追加するのでしょうか。それは、私たちが一貫してこの指標の重要性を確認してきたからです。実際、優れた運用パフォーマンスと組み合わされていない場合には、デリバリー パフォーマンスが組織パフォーマンスに悪影響を及ぼす可能性があることを示唆する証拠も得られています。従来のクラスタ分類手法とは異なり、この手法は、デリバリー パフォーマンスと運用パフォーマンスに共通するチームのパフォーマンスを表現しようとする、記述的試みです。したがって、どのクラスタがより優れているかは必ずしも明らかにはなりません。

まず、ソフトウェア デリバリーと運用パフォーマンスを理解するために使用している 5 つの指標の概要を見てみましょう。

ソフトウェア デリバリーと運用パフォーマンス

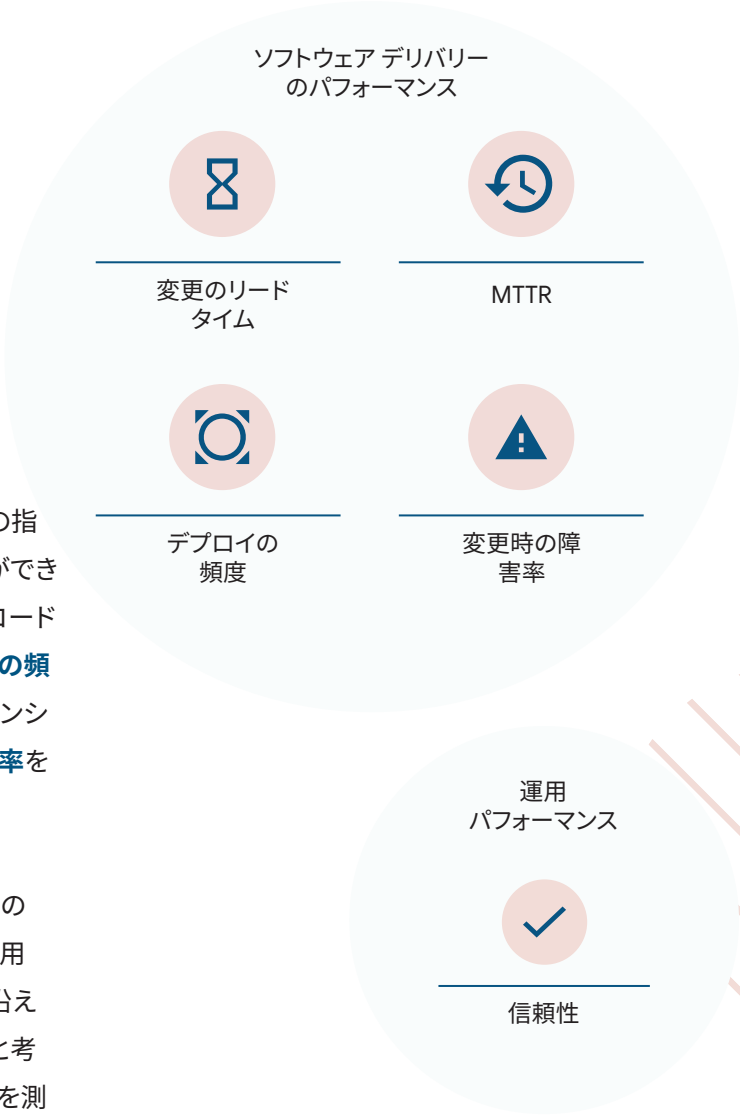
変化し続ける業界からの要求に対応するため、組織はソフトウェアを迅速に、高い信頼性で提供し、運用する必要があります。チームがソフトウェアへの変更を迅速に行えるほど、お客様に早く価値を提供し、実験を行い、貴重なフィードバックを得ることができます。私たちは 8 年間にわたってデータの収集と調査を行った結果、ソフトウェア デリバリーのパフォーマンスを測定するための 4 つの指標を作成し、検証しました。2018 年から、組織の能力を捕捉するため

5 つ目の指標も追加しました。5 つの基準すべてに秀でているチームは、組織として非常に優れたパフォーマンスを示しています。Google は、これら 5 つの基準を**ソフトウェア デリバリーと運用（SDO）のパフォーマンス**と呼んでいます。これらの指標はシステムレベルの成果に重点を置いたものです。機能が互いに相反する、ローカルな最適化により総合的な性能が低下するといった事態を招くことのあるソフトウェア指標追跡時の一般的な落とし穴を避けるうえで役立ちます。

デリバリー パフォーマンスと運用パフォーマンスの 5 つの指標

ソフトウェア デリバリーのパフォーマンスに関する 4 つの指標は、スループットと安定性に関するものと考えることができます。調査では、**コード変更のリードタイム**（すなわち、コードの commit から本番環境へのリリースまで）と**デプロイの頻度**を使用してスループットを測定しました。安定性は、インシデント発生後の**サービス復旧時間**と**変更時の障害発生率**を使用して測定しました。

5 番目の指標は**運用パフォーマンス**を表すもので、最新の運用プラクティスを測定します。私たちは、サービスが可用性やパフォーマンスなどのユーザーの期待にどれだけ沿えるかを示す**信頼性**が**運用パフォーマンス**の下地となると考えています。これまでの調査では信頼性ではなく可用性を測定しましたが、可用性は信頼性エンジニアリングに特有の指標であるため、2021 年に測定の幅を信頼性まで広げました。これにより、可用性、レイテンシ、パフォーマンス、スケーラビリティがより広く示されるようになります。具体的には、回答者に、信頼性に関する目標を満たす、または上回る能力を評価するように依頼しました。デリバリーのパフォーマンス レベルの異なる複数のチームで、デリバリーだけでなく運用パフォーマンスも重視する場合に、より優れた結果（燃え尽き症候群の減少など）を得られることが確認されました。



過去のクラスタ分類手法: デリバリー パフォーマンスのクラスタ分類

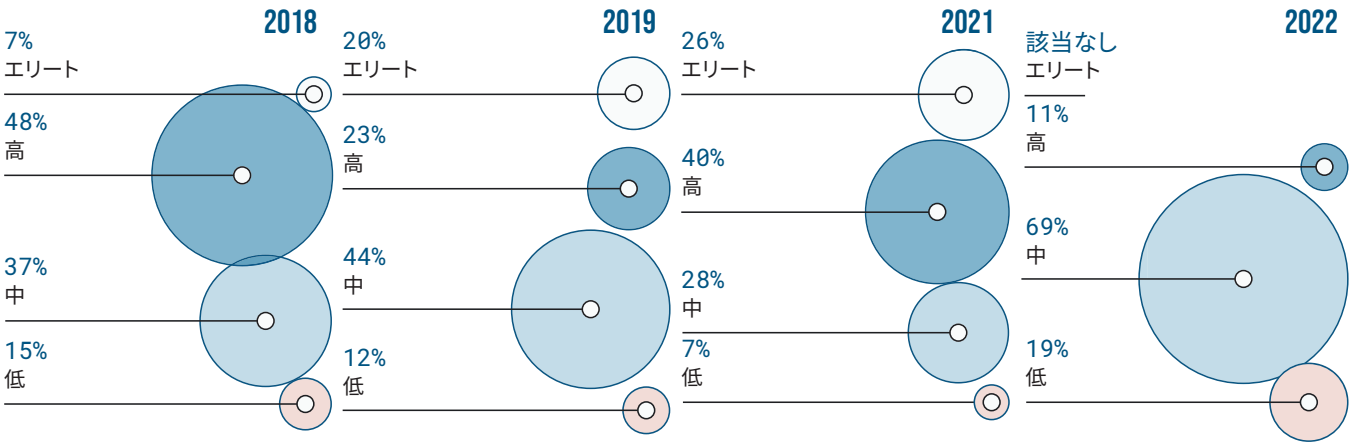
今年、2018 年から使用している 4 クラスタ ソリューションを評価したところ、データから、明らかに低パフォーマンスのクラスタと高パフォーマンスのクラスタがあることが確認できました。しかしながら、従来なら中程度のパフォーマンスと高いパフォーマンスを区別するために使用する 2 つのクラスタは、分割を必要とするほど差別化されていませんでした。さらに、適切なクラスタ ソリューションを選択するために使用するさまざまな指標では、適用するクラスタ分類手法にかかわらず、3 つのクラスタがデータを最も的確に捕らえることが例外なく示唆されました。下の表は、各クラスタのデリバリー パフォーマンスの特性について説明したものです。

昨年との顕著な違いとして、今年はどのクラスタもエリートとは考えていません。今年の高パフォーマンス クラスタでは、昨年の高パフォーマンス クラスタとエリート クラスタが混在しています。エリート クラスタを省くことにした理由は、最もパフォーマンスの高いクラスタが単に昨年のエリート クラスタの特徴を十分に示していないためです。

ソフトウェア デリバリーのパフォーマンス指標	低	中	高
デプロイの頻度 主なアプリケーションまたはサービスで、組織が本番環境にコードをデプロイする頻度、またはエンドユーザーにリリースする頻度はどのくらいか。	月 1 回から 6 か月に 1 回の間	週 1 回から月 1 回の間	必要に応じて (1 日に複数回のデプロイ)
変更のリードタイム 主なアプリケーションまたはサービスで、変更のリードタイム (commit されたコードが本番環境で正常に実行されるまでの時間) はどのくらいか。	1 か月から 6 か月の間	1 週間から 1 か月の間	1 日から 1 週間の間
サービス復旧時間 主なアプリケーションまたはサービスで、サービスのインシデントや、ユーザーに影響を与える障害 (計画外のサービス停止やサービス障害など) が発生した場合、サービスの復旧に通常どれくらいの時間がかかるか。	1 週間から 1 か月の間	1 日から 1 週間の間	1 日未満
変更時の障害率 主なアプリケーションまたはサービスで、本番環境に変更を加えた、またはユーザーに変更をリリースしたとき、サービスの低下 (サービス障害、サービスの停止など) が発生して、対策 (修正プログラム、ロールバック、フィックス フォワード、パッチなど) が必要になった割合はどのくらいか。	46%〜60%	16%〜30%	0%〜15%

このことは、このサンプルが、前進したと感じている従業員がいるチームや組織を代表していないことを示唆しています。一つの仮説として、現在のところ裏付けとなるデータがありませんが、ソフトウェア開発において、手法、ツール、情報共有の面でイノベーションが減少していることが挙げられます。これは、今も続くパンデミックの影響により、チームや組織間で知識や手法を共有することが難しくなった結果とも考えられます。互いに学び合う機会が少なくなり、その結果、イノベーションが鈍化した可能性があります。私たちは、この見解の背景にあるものをさらに深く掘り下げたいと考えています。

とはいえ、今年のパフォーマンスが低、中、高の各クラスタを昨年と比較すると、デリバリー パフォーマンスがやや高くなる方向に向かっていることがわかります。今年のクラスタは、昨年のクラスタの 2 つの間にあるように思われます。2022 年の高パフォーマンス クラスタは、2021 年の高パフォーマンス クラスタとエリート クラスタの間にあります。2022 年の低パフォーマンス クラスタは、2021 年の低パフォーマンス クラスタと中パフォーマンス クラスタの間にあるようです。低パフォーマンス クラスタが上方に向かっていることは、デリバリー パフォーマンスの上限が下がった一方で、下限が上がったことを示唆しています。



上表の割合内訳を見ると、パフォーマンスが高いクラスタの割合が 4 年ぶりの低水準にある一方で、パフォーマンスが低いクラスタの割合が 2021 年の 7% から今年は 19% へと大幅に上昇したことがわかります。今年の回答者は、3 分の 2 以上が中パフォーマンス クラスタに分類されます。高パフォーマンス クラスタとエリート クラスタが明らかに減少していることは、今年の回答者の多くが、現代の多くのチームで見られる DevOps 文化を確立していない、またはその確立の途上にある組織またはチームに所属していることを示唆しているとも考えられます。

2021 年と 2022 年の類似点より、違いを強調しすぎているかもしれません。2021 年と 2022 年のクラスタは、高パフォーマンス クラスタと低パフォーマンス クラスタの間に大きな隔たりがあるなど、多くの特徴を共有しています。たとえば、高パフォーマンス クラスタは、低パフォーマンス クラスタに比べて 417 倍のデプロイを行っていると推定されます。[1]

[1] この推定の根拠については、「手法」の項をご覧ください。

デリバリー パフォーマンスと運用パフォーマンスのクラスタ分類

私たちは、5 つの指標で表すことになっている**スループット**（コード変更のリードタイムとデプロイ頻度の複合）、**安定性**（サービス復旧時間と変更時の障害率の複合）、**運用パフォーマンス**（信頼性）の 3 つのカテゴリについて、クラスタ分析を行うことにしました。理由は、私たちのモデルにおいて、運用パフォーマンスが極めて重要な役割を果たすためです。

堅実な運用パフォーマンスを示さない組織の場合、スループットと安定性が組織パフォーマンスに与える影響は小さくなります。私たちは、運用パフォーマンスを考慮せずに DevOps パフォーマンスの背景を説明することは、全体像の重要部分を取りこぼすようなものだと考えています。

データを調査した結果、4 クラスタ ソリューションが導き出されました。以下は、4 つのクラスタの内訳とその名称です。

クラスタ	安定性		運用パフォーマンス	スループット		回答者の割合
	サービス復旧時間	変更時の障害率	信頼性	リードタイム	デプロイの頻度	
初期段階	1 日から 1 週間の間	31%〜45%	期待にかなう場合もある	1 週間から 1 か月の間	週 1 回から月 1 回の間	28%
フロー段階	1 時間未満	0%〜15%	たいてい期待にかなう	1 日未満	必要に応じて（1 日に複数回のデプロイ）	17%
後期段階	1 日未満	0%〜15%	たいてい期待にかなう	1 週間から 1 か月の間	週 1 回から月 1 回の間	34%
終了段階	1 か月から 6 か月の間	46%〜60%	たいてい期待にかなう	1 か月から 6 か月の間	月 1 回から 6 か月に 1 回の間	21%

ただし、同じクラスタに属する 2 つのチームを訪問したとしても、両者からはまったく異なる印象を受けることもあり、私たちのストーリーは、実際に見えるものを反映していないかもしれません。上記のクラスタごとに提示しているストーリーは、私たちが多くのチームと関わってきた経験を生かして、これらのデータ インテリジェンスのパターンを理解しやすくしようとするものです。さらに、同じチームを 2 種類の時点で訪問した場合、これらのチームが同じクラスタにとどまっていなかったこともあり得ます。この理由として考えられることとして、チームが改善または劣化したことが挙げられます。もう一つの可能性として、チームがそのアプリケーションまたはサービスの現状により適したデプロイ パターンに移行したことが考えられます。たとえば、アプリケーションまたはサービスの開発の初期段階では、チームは調査に重点を置き（初期段階クラスタ）、適所を見つけ始めると、信頼性に重点を移すことが考えられます（フロー段階クラスタまたは後期段階クラスタ）。

各クラスタには固有の特性があり、それぞれに大きな割合の回答者が属しています。

初期段階クラスタは、どの側面でもパフォーマンスが良くも悪くもありません。プロダクト、機能、またはサービスのデプロイの初期段階にある可能性があります。フィードバックの入手、プロダクト マーケット フィットの理解、より一般的な調査に焦点が当てられているため、信頼性にはそれほど重点が置かれていない可能性があります。

フロー段階クラスタは、高い信頼性、高い安定性、高いスループットのすべての特性で優れたパフォーマンスを発揮します。このフロー状態に達した回答者は、わずか 17% です。

後期段階クラスタの回答者は、あまり頻繁にデプロイしませんが、デプロイする場合は、成功する傾向にあります。回答の 3 分の 1 以上がこのクラスタに分類され、最大のクラスタとしてサンプルを最も代表するものになっています。このパターンは、段階的に向上しているチームに特有である傾向がありますが（しかし排他的ではない）、チームとその顧客は、アプリケーションまたはプロダクトの現状におおむね満足していると言えます。

最後に、**終了段階**クラスタは、チームやその顧客にとってまだ価値があるけれども、開発がもう積極的に進められていないサービスやアプリケーションに取り組んでいるチームのように見えます。

これらのクラスタには、その手法と技術的能力に顕著な違いがあります。私たちは、**フロー段階**クラスタが示す高度なソフトウェア デリバリーと運用パフォーマンスを考慮し、このクラスタが手法と技術的能力において他のクラスタとどのように異なるかを調査することにしました。その結果、他のクラスタと比較して、**フロー段階**クラスタが以下により注力していることがわかりました。

- 疎結合アーキテクチャ: 各チームが、他のチームが実施するシステム変更に依存することなく、自分たちのシステムの設計に大規模な変更を加えることができる程度
- 柔軟性の確保: 社員の勤務形態に関して会社がどれだけ柔軟であるか
- バージョン管理: アプリケーション コード、システム構成、アプリケーション構成などの変更を管理する方法
- 継続的インテグレーション(CI): ブランチをトランクに統合する頻度
- 継続的デリバリー(CD): 安全に、持続的に、効率的に変更を本番環境に反映するための機能

興味深いことに、**フロー段階**クラスタでは、ドキュメントをあまり重視しない傾向があります。昨年、私たちは、ドキュメント作成のプラクティスが、デリバリー パフォーマンスと運用パフォーマンスの両方(SDO)にとって重要であることを確認しました。では、なぜ**フロー段階**クラスタでは、ドキュメントを重視していないのに SDO が高いのでしょうか。ひとつには、ドキュメント以外にも高い SDO パフォーマンスを実現する手段が多数存在することが挙げられます。さらに、**フロー段階**クラスタは、コードのリファクタリングを継続的に行い、より自己文書化的なプロセスを作成することにより、アンケートで説明しているとおり、ドキュメントの必要性を減らしているとも考えられます。

最も多くの回答者が属する**後期段階**クラスタは、他のクラスタよりもクラウドネイティブ度が低い傾向にある大規模組織の回答者で構成される傾向にあります。成熟した企業で、プロセスがある程度固まっていて、最終的にはエンドユーザーに安定した信頼性の高い(価値のある)体験を提供する企業であることが想像できます。このクラスタは、パフォーマンス志向の、創造的な文化を示しています。[2] 後期段階クラスタの最も興味深い特徴のひとつとして、一般的な組み合わせではありませんが、スループットが低く、前向きに仕事に取り組む姿勢が強い(Westrum が言うところの「創造的な」労働文化)ことが挙げられます。スループットと労働文化が比例する(スループットが低く、前向きに仕事に取り組む姿勢が弱いか、スループットが高く、前向きに仕事に取り組む姿勢が強い)方が一般的です。今後、スループットと文化の関係をより深く理解するための研究を行いたいと考えています。

[2] (Westrum)

さらに、燃え尽き症候群、組織パフォーマンス、非計画的作業という 3 つの結果について、それぞれのクラスタの比較に焦点を当てました。その結果、私たちの予想を裏切る結果が得られました。**終了段階**クラスタが組織パフォーマンスにおいて他のクラスタを上回ったのです。このクラスタの特徴（安定性が低く、スループットが低い）を見ると、DORA のこれまでの調査結果の大部分と一見矛盾しているように見えます。しかし、この異常事態を偶然性のせいにするのではなく（その可能性は高いものの）、いくつか考えられる理由を探ってみたいと思います。

これらの事態を解明しようとするときに、補足として留意すべき見解があります。**終了段階**クラスタは、高い組織パフォーマンスを発揮するために、大きな犠牲を払っています。このクラスタのチームは、燃え尽き症候群に陥る割合が最も高く、最もミスが発生しやすいと感じており、最も計画性の低い作業を背負いがちです。同時に、これらの結果から、高い組織パフォーマンスを達成するには、信頼性があれば十分かもしれませんが、スピードと安定性がなければ、チームが燃え尽き症候群に陥り、非計画的作業を背負うという代償を払うことになることがわかります。

終了段階クラスタが他のクラスタ、特に**フロー段階**クラスタよりも組織パフォーマンスが高い理由を説明するための仮説が他にもあります。これを以下に示します。

- この 4 つのクラスタの根底には、私たちのデータにはない特徴があるとも考えられます。たとえば、組織の規模は成熟度を適切に表していると考えられます。**フロー段階**クラスタは、小規模な企業で構成される傾向があり、このことは、これら企業の製品がより発達段階にあることを示しているとも考えられます。

- また，**フロー段階**クラスタに属するのは小規模な企業であることが多く、過去のプロセスやインフラストラクチャにあまり縛られず、その結果、より高度な DevOps プロセスを導入していると考えられます。とはいえ、このデータは、テクノロジーとはほとんど無関係と考えられる理由で組織の規模が組織パフォーマンスと正の相関があることを示しています。

- **フロー段階**クラスタは、Westrum の創造的文化で説明されている原則を無視する傾向がありました。私たちは、このことが組織のパフォーマンスを低下させることが多いことを確認しました。

- 各クラスタの組織は、期待する信頼性とは何を意味し、どのようにこれを監視するのかについて異なる方針を持っていることが考えられます。また、組織パフォーマンス目標をどう定義するかについても、同様のことがいえます。

- **終了段階**クラスタは、短期的に高い組織パフォーマンスを発揮すると考えられますが、このクラスタはどのように長期的なパフォーマンスを発揮するのでしょうか。燃え尽き症候群が離職につながるのでしょうか。プロセスを調整できるのでしょうか。

- 私たちは、さまざまなレベルで質問を行っています。技術的能力（たとえば疎結合アーキテクチャ）についてチームのレベルで質問しています。組織パフォーマンスに関する質問は、組織レベルで行っています。組織には、たいてい多くのチームがあり、組織がうまく機能している一方で自身の所属チームがうまく機能していないと回答者が認識している可能性があります。

さらに、**フロー段階**クラスタでは、**終了段階**クラスタに次いで組織パフォーマンスが 2 番目に高く、燃え尽き症候群と非計画的作業の度合いが最も低い部類に入っています。**フロー段階**クラスタと**後期段階**クラスタの両方が示すとおり、DevOps の理念は、信頼性が確保されている場合に最も効果を発揮します。

引き続き、業界内のばらつきを説明できる新しい方法を探ってみたいと思います。さらに、このばらつきについて理解するうえで、運用パフォーマンスを関連項目として引き続き採用したいと考えています。また、高度に規範的で価値的なクラスタ（エリート クラスタなど）は避け、代わりに、一般的な SDO パフォーマンス群を特定するための、わかりやすい記述的試みに注力したいとも考えています。

03

どう改善するか

Derek DeBellis

さまざまな結果をどのように改善するか

State of DevOps レポートは、お客様のチームが、お客様の希望する成果を得るための DevOps の手法と能力に注力できるよう、事実に基づくガイダンスを示すものです。今年、私たちは、セキュリティとチームが求める一連の成果の両方についても調査を行いました。以前は、ソフトウェア デリバリーおよび運用パフォーマンス（SDO）と組織パフォーマンスの成果に焦点を当てていました。今もこれらに焦点を当てていますが、燃え尽き症候群、チームを推薦する可能性、非計画的作業、誤りの犯しやすさについても、SDO と組織パフォーマンスを改善するための手段としてだけでなく、これら自体も改善する目的で、調査したいと考えました。そのため、今年は、これらの成果に影響を与えると思われる手法と能力について言及することにしました。

今年、DORA の根底にある理論、すなわち「DevOps に万能の手法はない」をよりよく反映させるよう、調査モデルが変更されました。実際、私たちは、推奨事項を定めるには、より幅広くチームの背景を理解する必要があることに気付きました。あるチームにとって有益な手法が、別のチームにとっては有害な場合もあります。たとえば、私たちは長い間、技術的能力（疎結合アーキテクチャ、トランクベース開発、バージョン管理、継続的インテグレーションなど）は、継続的デリバリーが実施されている場合にソフトウェア デリバリー パフォーマンスにより顕著な好影響を与えるという仮説を立ててきました。今年、私たちは、この相互作用とその他の相互作用を明示的にモデル化しました。その目的は、単に「何が何に影響を与えるのか」から、「どのような条件下でこれらの影響が発生し、増幅され、あるいは減少するのか」といった疑問の答えを見つけられるよう、理解を深めることにあります。このような条件を理解することは、複雑で気の遠くなるような努力であることが証明されていますが、この初期の所見をいくつかを紹介したいと思います。

今年と前年までの調査モデルは、私たちのウェブサイトからオンラインでご覧いただけます。

4 つのキーを超えて

DORA の指標はどのように開発と運用パフォーマンスを向上させるのでしょうか。Liberty Mutual Insurance のソフトウェア エンジニアの部門横断チームは、DORA の「4 つの鍵」指標を使用して定期的にパフォーマンスを評価しています。たとえば、Liberty Mutual のシニア スクラム マスターである Jenna Dailey 氏は、あるチームが DORA の調査を活用してボトルネックを特定し、テスト駆動開発手法に移行して、全体のパフォーマンスを向上させるのに役立ったと語りました。

ソフトウェアの品質とデリバリーを改善するためにデータと DORA の指標を活用する Liberty Mutual の手法については、最近の Tomorrow Talks で詳しく紹介しています。

Eric Maxwell

クラウド

過去数年にわたり見てきた流れを基に、クラウド コンピューティングの利用が加速し続けています。実際、複数のクラウドを含むパブリック クラウドを利用している人の割合は、2021 年の 56% から 76% に増加しています。プライベート クラウドを含め、クラウドを一切利用していないと回答した人は、昨年の 21% からわずか 10.5% に減少しました。複数のパブリック クラウドの利用率は 21% から 26% に、ハイブリッド クラウドの利用は 25% 増の 42.5% に増加しました。また、プライベート クラウドの利用率は 32.5% と、昨年の報告である 29% から少し増加しました。

36%

パブリック クラウドの利用率が増加

25%

ハイブリッド クラウドの利用率が増加

クラウド コンピューティングの利用は、全体的な組織パフォーマンスに好影響を与えます。クラウドを使用している回答者は、クラウドを利用していない回答者と比べて、組織パフォーマンス目標を超える**傾向が 14% 高い**ことがわかりました。

前年までに示し、このレポートでも引き続き確認しているとおり、クラウド コンピューティングの利用は、全体的な組織パフォーマンスに好影響を与えます。クラウドを使用している回答者は、クラウドを利用していない回答者と比べて、組織パフォーマンス目標を超える**傾向が 14% 強い**ことがわかりました。私たちの調査から、クラウド コンピューティングによってチームがソフトウェアのサプライ チェーンのセキュリティや信頼性などを適切に確保できるようになり、それらが組織パフォーマンスに好影響を与えることがわかっています。

驚くべきことに、パブリック、プライベート、ハイブリッド、マルチなど、あらゆる種類のクラウドのユーザーは、変更時の障害率に対し負の関連性、つまり変更時の障害率が高くなることを示しました。これについては、さらなる調査が必要です。この理由を推測するのではなく、今後の研究でさらに詳しく調査することにします。しかし、わずかな例外を除いて、クラウドネイティブ アプリケーション（もともとクラウド用に設計、構築されたアプリケーション）の使用は、調査したすべての項目に対し好影響を示し、異彩を放っていました。

クラウド コンピューティング プラットフォームの利用は、パブリック、プライベートにかかわらず、企業文化と労働環境に関する成果に好影響（たとえば、創造的文化、燃え尽き症候群の減少、安定性の向上、従業員満足度の向上など）を与えます。クラウド ユーザーの方が、これらの文化面の成果に対して 16% 高いスコアを示しました。

前年と比べて上昇を続けるクラウド利用率

	2022	2021 年比
ハイブリッド クラウド	42.47%	25%
1 つおよび／または複数のパブリック クラウド	76.08%	36%
プライベート クラウド	32.55%	12%
クラウド利用なし	10.55%	-50%

ハイブリッド クラウドとマルチクラウド（およびプライベート クラウド）の利用は、回答者に高度の**信頼性**が伴う**場合を除き**、ソフトウェア デリバリー パフォーマンスの指標（MTTR、リードタイム、デプロイ頻度）に**負の**影響を及ぼすように見えます。

組織パフォーマンスを向上させるハイブリッド クラウドとマルチクラウド

ハイブリッド クラウドや複数のパブリック クラウドの利用が組織にプラスの影響を与えるという強い兆候が引き続き確認できます。複数のクラウドを使用する実務者は、クラウドを使用しない実務者と比較して、組織パフォーマンスが 1.4 倍高くなることがわかりました。ただし、ハイブリッド クラウドやマルチクラウド（プライベート クラウドも含む）の利用は、回答者の信頼性が高くない限り、複数のソフトウェア デリバリー パフォーマンスの指標（MTTR、リードタイム、デプロイ頻度）に負の影響を与えるように思われます。このことは、堅牢な SRE 手法と、ソフトウェア デリバリーにおいて信頼性が果たす役割の重要性を如実に物語っています。

2021 年には、複数のパブリック クラウドを利用する主な理由を回答者に尋ねましたが、2022 年には、複数のクラウド プロバイダを利用することによって得られるすべての利点を回答者に尋ねました。最も多く報告された利点は「可用性」でした。これは、業界における信頼性についての注目度やフォーカスとも符合します。可用性が確保されていなければ信頼性があるサービスとはいえません。50% 以上の実務者が、複数のクラウド プロバイダが持つ独自の利点を活用していると回答しました。

複数のクラウド プロバイダを採用する利点

可用性	62.61%
各プロバイダ固有の利点の活用	51.59%
複数のプロバイダへのリスク分散	47.54%
障害復旧	43.48%
法令遵守	37.97%
交渉戦術または調達要件	19.13%
その他	4.06%

>50%

複数のクラウド プロバイダを利用している回答者の割合

クラウド コンピューティングの 5 つの特性を利用することは、組織パフォーマンス向上につながる長期的な好循環への重要な第一歩です。

クラウド コンピューティングの 5 つの特性

前回の調査手法と同様に、参加者がクラウド コンピューティング技術を利用しているかどうかだけでなく、クラウド コンピューティング技術をどのように使用しているかを確認することを目的としました。このために、米国国立標準技術研究所（NIST）が定義するクラウド コンピューティングの 5 つの基本特性について質問しました。

オンデマンド セルフサービス - ユーザーは、必要に応じてコンピューティング リソースを自動的にプロビジョニングでき、プロバイダ側で人手による介入は必要ありません。

広範なネットワーク アクセス - 幅広い方法で機能を利用でき、ユーザーはスマートフォン、タブレット、ノートパソコン、ワークステーションなど、複数のクライアントから機能にアクセスできます。

リソースプール - プロバイダのリソースはマルチテナント モデルでプールされ、物理および仮想リソースがオンデマンドで動的に割り当ておよび再割り当てされます。

顧客は一般に、提供されるリソースの正確な場所を直接コントロールできませんが、より高レベルの抽象化で、たとえば国、州、データセンターなどの場所を指定できます。

迅速な弾力性 - 能力は弾力的にプロビジョニングしてリリースし、必要に応じて迅速にスケールアウトまたはスケールインできます。ユーザーからは、無制限の能力がプロビジョニング可能で、いつでもどれだけでも充当できるように見えます。

測定サービス - クラウド システムは、サービスのタイプに適切なレベルの抽象化、たとえばストレージ、処理、帯域幅、アクティブなユーザー アカウントなどで測定機能を利用し、リソースの使用を自動的にコントロールして最適化できます。リソースの使用をモニタリング、コントロール、報告することで透明性を実現できます。

このレポートでは、過去 3 年間の DORA の研究を検証し、組織内にこれら 5 つの特性が存在すると、ソフトウェア デリバリーと運用パフォーマンスに好影響を与えると結論付けています。また、組織に好影響を与えるプロセスを導入することで、これらの特性が組織パフォーマンス向上につながることもわかりました。クラウド コンピューティングの 5 つの特性を示すことは、組織パフォーマンス向上につながる長い道のりの第一歩となります。

2022 年現在、クラウド コンピューティングの差別化要因を活用するチームがますます増えてきています。4 年連続で、クラウド コンピューティングの 5 つの特性の採用が増えています。リソースプールは 14% 増と最大の伸びを示し、昨年 2 番目に多く利用された迅速な弾力性は 5% 増と伸びとしては最小になりました。

NIST	2021	2022	変化率
広範なネットワーク アクセス	74%	80%	8
迅速な弾力性	77%	81%	5
オンデマンド、セルフサービス	73%	78%	7
メジャード サービス	78%	83%	7
リソースプール	73%	83%	14

14%

リソースプールの増加

Dave Stanke

SRE と DevOps

優秀な技術チームは、コードを出荷、さらに言えば高品質のコードを出荷すること以外でも組織に貢献します。また、技術チームが提供するサービスが長期にわたって利用可能で、優れており、時間の経過とともに他の部分でもユーザーの期待に沿うことも保証してくれます。信頼性は、チームがこれらの取り組みをどの程度守っているかを示す多面的な指標であり、今年も引き続き、ソフトウェア デリバリーと運用における要因としての信頼性について探求しました。

サイト信頼性エンジニアリング（SRE）は、Google 発で、現在では多くの組織で実践されている、影響力のある運用手法です。SRE では、経験則に基づく学習、部門横断的な協力、自動化の積極的利用、サービスレベル目標（SLO）などの測定技法の使用を重視します。他の最新の運用手法でも同様の方法を採用していますが、異なる命名規則を適用しています。そのため、私たちの調査では、これらの手法の程度をできるだけ客観的に評価するために、回答者に提示する文章には中立的で、説明的な言葉を使用するように配慮しています。また、信頼性エンジニアリングの成果、すなわちチームが信頼性目標を達成できる程度についてもデータを収集しています。SRE の手法と信頼性の成果という入力と出力の両方が、他の DevOps 機能とともに予測モデルに反映されています。

信頼性が不可欠

SRE の採用は私たちが調査したチームに広く浸透しており、回答者の過半数が、今回問い合わせた手法のうち 1 つ以上を利用しています。調査したチーム全体のデータから、信頼性、ソフトウェア デリバリー、成果の間の微妙な関係がわかります。つまり、信頼性が低い場合、ソフトウェア デリバリー パフォーマンスが組織の成功を予測する要因とはなりません。しかしながら、信頼性が向上すると、ソフトウェア デリバリーがビジネスの成功に好影響を与えることがわかります。

> 信頼性がなければ、ソフトウェア デリバリー パフォーマンスが組織の成功を予測する要因とはなりません。

SRE への投資により信頼性が向上しますが、そのためには、採用が一定程度に達している必要があります。

この現象は、SRE「エラー バジェット」フレームワークを利用する場合にも当てはまります。つまり、サービスが信頼できない場合、ユーザーは、コードをその脆弱な状況に迅速に push できても、メリットを得られません。

サイト信頼性エンジニアが長年主張してきたとおり、信頼性は、あらゆるプロダクトの最も重要な「機能」です。私たちの調査により、ユーザーへの約束を果たすことが、ソフトウェア デリバリーの改善により組織に利益をもたらすために必要な条件である、という見解が裏付けられます。

しかし、このような困難を乗り越えた企業は、多くの場合、継続的に高いレベルの業績を達成します。

今年の調査では、調査対象となった技術系チーム全体に J カーブのパターンが見られることがわかっています。信頼性エンジニアリングの実践が少ない場合（SRE 導入の初期段階であることを示唆）は、この実践から信頼性が向上するとは予測できません。しかし、SRE の導入が進むにつれて、SRE の利用が信頼性、ひいては組織パフォーマンスを正確に予測できるようになる転換点に到達します。

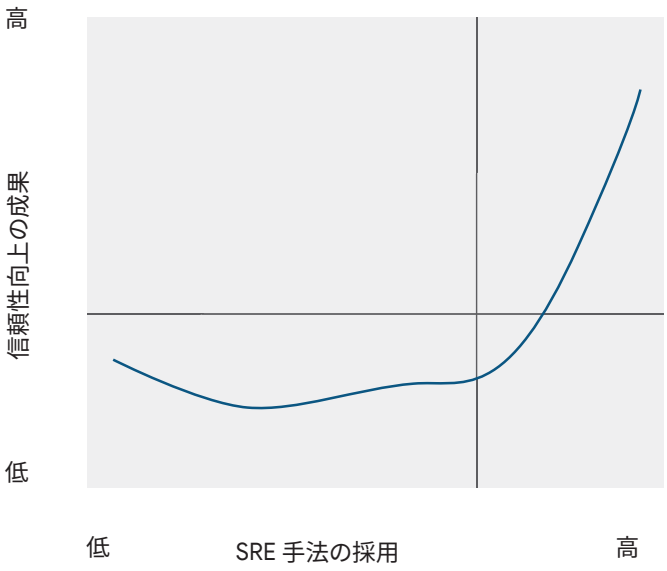
J カーブを認識する

信頼性の確立に向けては、どのような課題が待ち受けているのでしょうか。DORA の調査協力者である James Brookbank と Steve McGhee は、O'Reilly 出版の『Enterprise Roadmap to SRE』[1] において、既存の組織で SRE を導入した経験を振り返り、「変化の J カーブを認識すること」を推奨しています。2018 年の State of DevOps レポートで紹介した「J カーブ」は、組織の変革が初期に実を結び、その後成果が減少または後退する時期が来る傾向がある現象です。

[1] https://sre.google/resources/practices-and-processes/enterprise-roadmap-to-sre/

信頼できるチームが信頼できるサービスを提供し、創造的なチーム文化の有無から信頼性向上の有無を予測できます。

SRE を実践するための初期段階にあるチームは、途中でつまずくことを覚悟しておく必要があります。文化、プロセス、ツールのすべてが新たな指針に沿って再編成されるため、長い道のりになることがあります。しかし、時間をかけて継続的に投資すれば、成功する可能性があることは確かです。



SRE 導入の初期段階から粘り強く努力するチームは、信頼性が大きく向上する

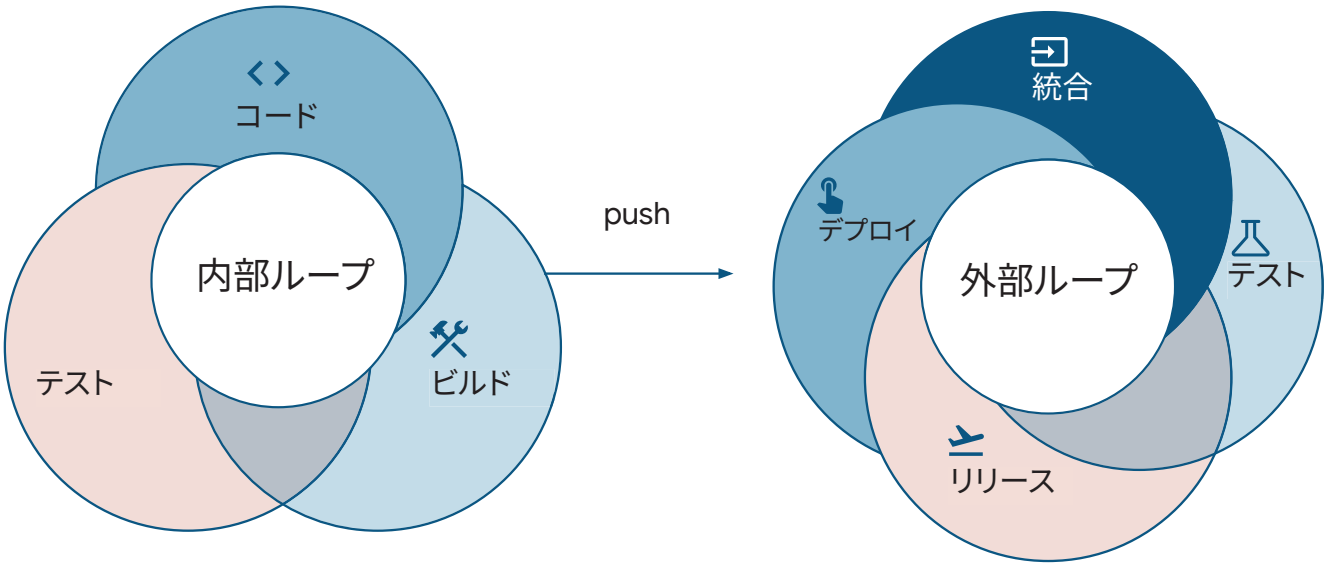
人、プロセス、ツールへの投資

信頼性とは人間の努力で得られるものであり、SRE 手法により、多くの形でこれが例証されます。SRE の基本原則の一つとして、内部監視データとは対照的に、ユーザーの認識こそが信頼性を測る真の尺度であることが挙げられます。そのため、信頼性がチーム内の肯定的な力学によって促進されるのは当然のことかもしれません。私たちは、信頼と協調を示す「創造的な」文化を持つチームほど、SRE を実践し、信頼性向上の優れた成果を上げる可能性が高いことを確認しました。また、メンバー構成が常に変わらない安定したチームは、ユーザー向けサービスにおいてもより高い信頼性を確保できます。また、DevOps 全体と同様に、信頼性エンジニアリングの取り組みにおいても、プロセスとツールによって人による作業を補強することが有効となります。クラウド コンピューティングと継続的インテグレーションの利用などの取り組みにより、信頼性が向上することが予測できます。

Eric Maxwell

技術面の DevOps 機能

今年、私たちは、さまざまな技術的手法によって導かれる成果を把握するため、さまざまな技術的能力を検証しました。コード記述、テスト、バージョン管理への push などの開発者による作業から成る「内部ループ」と、コードマージ、自動コードレビュー、テスト実行、デプロイ、リリースなどの活動を含む「外部ループ」の、ソフトウェア開発の 2 大段階について検討しました。



信頼性目標を達成している高パフォーマーは、**CI を使用する傾向が 1.4 倍強い。**

私たちの調査によると、内部ループと外部ループの開発に優れた企業は、より速く、より確実にコードを出荷できます。高パフォーマンスの実現に最も寄与する機能は、バージョン管理、継続的インテグレーション、継続的デリバリー、疎結合アーキテクチャです。

信頼性目標を達成している高パフォーマー:

33% バージョン管理を実施する傾向がより強い

39% 継続的インテグレーションを実施する傾向がより強い

46% 継続的デリバリーを実施する傾向がより強い

40% 疎結合アーキテクチャに基づくシステムを導入する傾向がより強い

実際に、平均以上に上記の機能をすべて利用している回答者は、これらの機能を利用していない回答者より**組織パフォーマンスが 3.8 倍高くなっています**。

継続的インテグレーション

継続的インテグレーションは、CI と呼ばれることも多く、外部ループ開発プロセスの一部であり、コードのデプロイ準備が完了しているかを判断するために、コード commit の度に自動的にアーティファクトをビルドし、一連の自動テストを実行します。このプロセスにより、開発者が迅速かつ自動的にフィードバックを提供し、より高い信頼感を持って作業できます。CI は、開発者のワークステーションから本番環境にコードを移行するための重要な要素です。例年どおり、CI はデリバリー パフォーマンスを促進することがわかっています。**信頼性目標を達成している高パフォーマーは、CI を使用する確率が他の人の 1.4 倍高くなっています。**

今年は、外部ループ開発プロセスのもう一つの部分である継続的デリバリーについて、もう少し深く掘り下げましたが、これについては後の章で述べます。しかし、まずは、継続的インテグレーションの補完的な要素であるトランクベース開発について見てみましょう。

トランクベース開発

トランクベース開発とは、コードを継続的にトランクにマージし、機能ブランチの長期化を避ける手法です。この手法は、継続的インテグレーションを補完するものと考えられており、ソフトウェア デリバリーの速度を引き上げることが長年にわたって証明されてきました。

今年は、業務経験年数に関するユーザー属性が変化したため、トランクベース開発を実施する場合は経験が重要であることが確認できます。昨年は、回答者の 40% が 16 年以上勤務していると答えましたが、今年は 13% にとどまりました。引き続き「デリバリーは状況に左右される」というテーマを検証すると、全体的に経験の浅い人は、トランクベース開発に関して、以下のとおりあまり良い結果を得られないことがわかります。

▼ ソフトウェア デリバリーの全体的なパフォーマンスが**低下する**
▲ 計画外作業の量が**増加する**
▲ 誤りを**犯しやすくなる**
▲ 変更時の障害率が**上昇する**

16 年以上の経験を持ち、トランクベース開発を行っている人は、その手法の利点を実感し、以下のような結果を得ています。

▲ ソフトウェア デリバリーの全体的なパフォーマンスが**向上する**
▼ 計画外作業の量が**減少する**
▼ 誤りを**犯しにくくなる**
▼ 変更時の障害率が**低下する**

これは、トランクベース開発をうまく導入するために必要な追加の手法によるものだと考えられます。壊れたトランクを絶対に放置しないというルールを厳格に適用していないチームや、ゲート型コードブランチやトランクを破損するコードの自動ロールバックを利用していないチームは、トランクで開発しようとすると確実に苦労します。

しかしながら、トランクベース開発を採用することで、組織全体のパフォーマンスに好影響が与えられます。

Frank Xu

継続的デリバリー

継続的デリバリー（CD）とは、以下の特徴を示すソフトウェア開発手法です。

1. チームがソフトウェアをいつでも本番環境またはエンドユーザーにデプロイできるようになる。
2. 新機能についての作業時を含め、ソフトウェアがそのライフサイクル中にデプロイ可能な状態になるようにする。
3. チームがシステムの品質とデプロイ可能性を確認できるようにし、デプロイを遮断して問題解決を優先する高速フィードバック ループを確立する。

なお、継続的デリバリーは、必ずしも継続的デプロイ（各ソフトウェア ビルドが自動的にデプロイされる手法）を意味しません。継続的デリバリーには、ソフトウェア ビルドをいつでもデプロイできることだけが求められます。

昨年は、チームが CD を実践する可能性を予測する技術的な DevOps 能力を調査し、疎結合アーキテクチャと継続的テスト、継続的インテグレーションなどの要因が最も強い予測因子であることを確認しました。今年は、CD の採用を促進する要因の調査に加え、CD 単独、および他の DevOps 機能との相互作用が開発の成果に及ぼす影響を分析、特定しました。

CD がソフトウェア デリバリー パフォーマンスを向上させる

前年までの所見と同様、CD の採用は、単独での採用の場合も他の DevOps 機能との併用の場合も、ソフトウェア デリバリー パフォーマンス向上の予測因子となります。CD を熱心に採用するチームほど、コードを本番環境にデプロイする頻度が高く、変更とサービス復旧のリードタイムが短くなる傾向が強まります。

バージョン管理と継続的デリバリーを合わせて採用しているチームは、どちらか一方のみ採用しているチームより 2.5 倍高いソフトウェア デリバリー パフォーマンスを発揮する傾向にあります。

また、バージョン管理も採用しているチームの回答者は、ソフトウェア デリバリー パフォーマンスが向上したと報告した人がこれを採用していないチームの回答者より 2.5 倍多く見られました。

CD により計画外作業が増える可能性あり

データから、継続的デリバリーの採用により、開発者がやり直し作業と計画外作業に費やす時間が増えることがわかりました。この結果に対する理由として、フィードバック ループが緊密さを増すと、開発者が繰り返しアプリケーションを構築することになる傾向が強まることが想定されます。結果として、開発者が反復的な変更をシステムの同じ部分に対する計画外作業と捉えることがあります。この作業は、計画外であるのと同時に、前回のデプロイからのフィードバックによって推進されることもあります。

技術的手法と CD

私たちの調査により、幅広い技術的機能が CD を支えていることが定期的に示されてきました。今年、私たちは、これらの個々の機能が CD と併用された場合にどうなるかを調査しました。その結果、トランクベース開発と疎結合アーキテクチャは、CD と併用することで、チームのパフォーマンスに負の影響を及ぼす可能性があることがわかりました。たとえば、疎結合アーキテクチャと CD を一緒に採用しているチームは、CD だけを採用しているチームと比較して、平均以上に誤りが起こりやすい（つまり、サービス停止、セキュリティ脆弱性、サービスの重大なパフォーマンス低下）と予測する傾向が 43% 強いことが確認できます。これらの影響についてはさらに調査する必要があり、改善中のチームにとってなんらかの摩擦が生じている可能性があることが示唆されています。この摩擦は、変革の J カーブと関係がある可能性があります。変革の J カーブでは、チームが早期に改善を実現しますが、比較的容易な目標に到達後は、勢いが衰えてしまいます。改善の可能性を最大限高めるには、改善に対する真摯な取り組みが必要です。CD のような機能を向上させる場合、チームと全体的パフォーマンスへの影響を注視する必要があります。

David Farley

疎結合アーキテクチャ

疎結合システムは、チームと組織が有効に機能するために重要です。これは、クラウドベースのシステムやマイクロサービスベースのシステムだけに当てはまることではなく、組織の変更能力にも関係しています。組織がソフトウェアを安全に、安心して変更できるかどうかが、そのソフトウェアの品質を示す指標となります。

疎結合アーキテクチャでは、以下のことが可能です。

- システムに変更を加えるために他のチームに依存する必要なく、自分たちのシステムの設計に大規模な変更を加える
- 独立したオンデマンドのテストにより、より少ない調整費用で、より迅速なフィードバックを得る
- ごくわずかなダウンタイムでコードをデプロイする

今年のレポートでは、構築するソフトウェアが疎結合アーキテクチャを採用しているかどうかを回答者に尋ねました。その結果は興味深いもので、疎結合アーキテクチャの存在とチームのパフォーマンスとの間には、複数の項目にわたって、ほぼ正のさまざまな相関関係があることがわかりました。

疎結合アーキテクチャの利点

疎結合アーキテクチャに沿ってソフトウェアを構築することを重視しているチームは、高い安定性、信頼性、スループットを発揮しやすくなります。こういったチームは、自分たちの職場を友人または同僚に推奨する傾向も強いとされます。

クラウドにデプロイしてマイクロサービス アーキテクチャ手法を採用し、何百ものサービスを管理しているチームで、疎結合アーキテクチャを採用しているソフトウェアを見かけることがよくあります。

疎結合アーキテクチャに沿ってソフトウェアを構築することを重視しているチームは、高い安定性、信頼性、スループットを発揮しやすくなります。

managing hundreds of services.ただし、疎結合は、1 つのシステム内のサービス数を表す単純な指標以上のものです。疎結合アーキテクチャの構成要素は、個別にデプロイできます。この独立性により、チームは、費用がかかるチーム間の調整オーバーヘッドなしに、サービスの開発、テスト、デプロイを行えます。

現実には、疎結合は 1 つのアーキテクチャ様式に限定されるものではなく、基本的には、システムのある部分に変更を加えても、その変更が他の部分に影響を与えない能力を意味します。これにより、組織は作業を分担することで、個々のチームが他のチームと調整することなく作業を進めることができます。

私たちの経験では、ソフトウェアのデプロイ前にその信頼性を高めるために他のサービスとの詳細な統合テストを必要とするチームは、まだ疎結合を実現できていません。疎結合の実現には、システム間のインターフェースと分離を改善することが推奨されます。インターフェースと分離を改善する効果的な方法の一つとして、サービスと構成要素の「テストの容易さ」を向上させることが挙げられます。設計によってサービスを分離してテストすることができれば、そのインターフェースは定義上、疎結合といえます。

また、疎結合アーキテクチャを採用している、団結力のある安定したチームは、継続的な改善を奨励、支援するソフトウェア開発手法を採用する傾向が強いことがわかりました。たとえば、信頼性目標を設定して作業の優先順位付けを行ったり、定期的なレビューを行って証拠に基づき信頼性目標を修正したりといった SRE の手法は，いずれも疎結合アーキテクチャを支えるものです。

また、疎結合アーキテクチャでは、他のチームとの調整が不要な独立したチームが自由にチームの規模を独自に拡大できるため、組織として従業員をより簡単に増員できます。

つまり、ソフトウェア サービスの疎結合が与える影響は、技術的な影響だけではありません。ソフトウェア開発の社会技術的な側面にも影響を与えます。結合は、コンウェイの法則（組織の設計システムはその組織のコミュニケーション構造を反映するという考え）の根底にあるものです。システムの結合が疎であるほど、組織も開発手法がより分散的でスケーラブルな、より結合が疎な組織といえます。

予想外の発見

今年の調査では、疎結合アーキテクチャがチームの燃え尽き症候群に影響している可能性があることが明らかになりました。これは、前年までの調査結果とは矛盾する予想外の発見です。私たちの分析では、情報が自由に行き交う安定したチームでは、燃え尽き症候群の程度が低いことがわかっています。Westrum の創造的文化とチームの安定性はどちらも疎結合アーキテクチャを支え、燃え尽き症候群を減少させるため、これは明らかに矛盾しています。確定的な結論を出すには、さらなる調査が必要です。

同時に、統合されたセキュリティ組織によってセキュリティ要件が定義、管理されている場合、チームが自分たちのソフトウェアを他のチームから切り離すことがより困難となることがあります。これは、アプリケーションに最も責任を負うチームにセキュリティの問題を移管する利点をさらに実証するものです（「サプライ チェーン セキュリティが重要な理由」もご覧ください）。これは、組織におけるより捉えにくい結合の形態の一つであり、今回はセキュリティに関するデータを収集しましたが、他の一元管理型機能にも同様に当てはまる可能性があります。セキュリティやその他のよく一元化される機能について各チームが独自に判断できるようにすることは、疎結合アーキテクチャを採用することで組織にもたらされる利益を享受するための一つの方法です。

Daniella Villalba

文化

「こちらではこうするんですよ。」

さまざまな業界の人々が、課題や機会に対する組織の取り組みを表現するために、数え切れないほどこの言葉を口にしてきたのではないでしょうか。

どの組織にも独自の文化があり、私たちの調査では、文化が組織の成功と従業員の幸福の基礎となることが一貫して示されています。

最も基本的なレベルでは、DevOps は、ソフトウェアを迅速、確実、かつ安全に開発し提供するためのツール、手法、および**人々の協力方法**に関するものであるため、文化も DevOps に必要な側面といえます。ある組織の文化に影響を与える要因を理解することは、リーダーが文化に関連する課題に正面から取り組むのに役立ちます。そのため、健全な文化を育むことを組織の優先事項とする必要があります。このような文化に関連する課題を放置しておくと、DevOps の手法が定着しづらくなる可能性があります。

今年も引き続き、Westrum の組織類型論を用いて、ある組織の文化の健全性を測定しました。さらに、チームの離脱、柔軟な勤務形態、認識された組織の賛同、燃え尽き症候群を測定することで、文化に関する理解を深めました。

今年の調査で得られたデータは、**組織パフォーマンス**が組織内に存在する文化の形態に影響されるというこれまでの調査結果を裏付けるものでした。特に、**創造的な文化**は、官僚的文化または不健全な文化を特徴とする組織と比較して、より高度な組織パフォーマンスと関連付けられます。

pathological culture.創造的な文化を持つ組織の従業員は、安定したチームに属し、より質の高い文書を作成し、有意義な仕事に大部分の時間を費やす傾向があります。

チームの離脱

私たちは、チームの離脱について調査し、**安定したチーム**、すなわちメンバー構成がこの 12 か月間ほとんど変わらなかったチームの方が高パフォーマンス組織に属している傾向が強いことを確認しました。絶えずチームからメンバーが離脱すると、新しいメンバーがチームに慣れるには時間が必要なため、生産性と勤労意欲に影響が出る場合があります。And those who stay

Westrum の組織類型

不健全な組織 権力志向	**官僚的な組織** ルール指向	**創造的な組織** パフォーマンス指向
協力的でない	積極的な協力ではない	積極的に協力
悪い知らせを伝えた人が「非難」される	悪い知らせを伝えた人が無視される	悪い知らせを伝えるよう教育される
無責任	責任範囲を限定	責任を分担
仲介を阻止	仲介を許容	仲介を奨励
失敗を責任転嫁	失敗は制裁	失敗を調査
新規性を否定	新規性を問題視	新規性を積極的に受け入れる

そして、チームにとどまる人は、業務量とチームの力学の変化に適応することが求められる場合もあります。また、調査により、メンバーが安定したチームの方がメンバー変更の多いチームより質の高いドキュメントを作成している傾向が強いことがわかりました。絶えず変化せざるを得ないチームは、質の高いドキュメントを作成できるようなやり方を継続することがより困難であるとも考えられます。

パフォーマンスの高い組織は、柔軟な業務形態を取り入れている傾向が強くなっています。

柔軟な業務形態

COVID-19（新型コロナウイルス感染症）の流行以降、多くの組織が柔軟な勤務形態を採用していることから、従業員にリモートワーク、出社、またはその両方を織り交ぜた働き方から自由に選択させることが組織パフォーマンスの向上につながるかを調査しました。

in-person, or hybrid options was associated with higher organizational performance.その結果、**従業員がより柔軟に業務形態を選べる**組織は、業務形態がより厳格な組織と比較して、組織パフォーマンスが高いことがわかりました。この結果から、従業員に必要に応じて自由に勤務形態を変更させることは、組織に具体的かつ直接的な利益をもたらすことがわかります。

燃え尽き症候群

燃え尽き症候群とは、仕事に関して気が進まない、興味がわかない、馬鹿らしいと感じる状態をいいます。燃え尽き症候群を経験すると、やる気がなくなって疲れるだけでなく、仕事に対する満足度が低下し、離職につながることもあります。燃え尽き症候群は、うつ病や不安神経症のリスク増加、心臓病、希死念慮など、幅広い心理的、身体的な健康状態の悪化と関連しています[1]。

昨年、私たちは、COVID-19 が流行する中で燃え尽き症候群を測定し、創造的文化が従業員の燃え尽き症候群発生の割合の低さと関連していることを確認しました。

[1] Maslach C, Leiter MP.Understanding the burnout experience: recent research and its implications for psychiatry（英語）。World Psychiatry。2016 Jun;15（2）: 103〜11. doi: 10.1002/wps.20311。PMID: 27265691; PMCID: PMC4911781。

柔軟な業務形態の採用により、従業員の燃え尽き症候群が減少し、従業員が所属チームを良い職場として推奨する傾向が強まります。

今年、私たちはあらためてこのことを確認し、チームメンバーの安定と柔軟な勤務形態が燃え尽き症候群の減少と関連していることを示すことで、燃え尽き症候群に関する理解を深めました。さらに、今年はチームのネット プロモーター スコア (NPS) を測定しました。これは、人々が自分のチームを友人や同僚に薦めるかどうかを示す指標です。結果として、チームの NPS は、上層部による賛同と関連していることがわかりました。また、燃え尽き症候群の結果と同様に、創造的文化、チームメンバーの安定、柔軟な勤務形態は、自分のチームを他人に薦める傾向の強さと関連していることもわかりました。

従業員が所属組織をどうとらえているか

最後に、今後 12 か月で自分のチームがどれだけの支援を受けられるかを予測するよう回答者に依頼することで、上層部の賛同の認識について調査しました。その結果、上層部から賛同を得られているという認識 (たとえば、財政的支援、リソースの割り当て、後援が多く得られているなど) が強いほど、組織パフォーマンスが高いことがわかりました。

また、今後 12 か月間にセキュリティ侵害か完全なサービス停止が発生する可能性を回答者に予測してもらいました。その結果、パフォーマンスの高い組織で働く人々は、重大な誤りが発生することを予測する傾向が低く、所属組織に対してより肯定的な見通しを持っていることがわかりました。同様に、ソフトウェアとデリバリーのパフォーマンスが高い組織で働く人々は、ビジネス面の成果を向上させるために現在の手法を変える必要があると感じる**傾向が低い**ことがわかりました。

社会的出自との関連について

調査の結果、パフォーマンスの高い組織か低い組織かにかかわらず、社会的地位の低いグループに属する従業員ほど、計画外の作業に多くの時間を費やしている傾向が強いことがわかりました。また、社会的地位の低いグループに属する従業員は、社会的地位の低いグループに属さない従業員と比較して、燃え尽き症候群の程度が高いこともわかりました。チームリーダーは、業務量の不均衡のリスクを認識し、チームメンバー間で公平に業務が配分されているかを確認する必要があります。

これらの結果を総合すると、組織レベルとチームレベルの両方で、従業員にとって健全でインクルーシブな環境を作ることが重要であることが見えてきます。

私たちは、文化の重要性を引き続き強調する一方で、組織の文化を変えること、または改善することが容易ではないことを認識しています。組織がまず従業員の経験を理解してから、DevOps 変革の取り組みの一環として、文化に関連する問題への対処にリソースを投入するよう努めることをおすすめします。

04

サプライ チェーン セキュリティが重要な理由

John Speed Meyers 氏

Todd Kulesza

2020 年 11 月、ソフトウェア サプライ チェーンのセキュリティ危機が起こりつつあると考えていた専門技術者はかなりの少数派でした。オープンソース ソフトウェアのセキュリティに注力できるよう、過去の取り組みを引き継ぐ Open Source Security Foundation が設立されており、この問題への取り組みに向けて明るい話題もいくつかありましたが、大手新聞の一面を飾るほどの話題ではありませんでした。大規模な攻撃 SolarWinds が

そのすべてを変えました。攻撃者がトロイの木馬ソフトウェアのアップデートを背景に、何千もの大手企業や政府のネットワークに静かに侵入できるようになったことで、時代が急速に変化しています。

今日、ソフトウェア サプライ チェーンのセキュリティの話題は、経営幹部の中で確実に、緊急課題として広く認識されるようになっています。

多くの取り組みが進められており、ソフトウェア業界の大部分は、自社のソフトウェア サプライ チェーンのセキュリティ対策を刷新し、オープンソース領域のセキュリティを向上させることを約束しています。

この章では、2 つの取り組みに焦点を当てます。1 つはソフトウェア アーティファクトのサプライ チェーン レベル（SLSA、「サルサ」と発音）、もう 1 つは NIST の Secure Software Development Framework（SSDF）です。それぞれ、攻撃者がソフトウェア作成プロセスを改ざんしたり、悪意のあるソフトウェア更新によってネットワーク防御者の目をすり抜けることができないようにするための、さまざまな防御策を提示しています。

では、SLSA と SSDF に関連するソフトウェア サプライ チェーンのセキュリティ対策は、どの程度普及しているのでしょうか。どのセキュリティ対策について採用促進に支援が必要で、どのセキュリティ対策がすでに広く使用されているのでしょうか。これまで、これらの質問に対する理路整然とした回答はありませんでした。何百人ものソフトウェア専門家を対象に、サプライ チェーン セキュリティに関連する対策の導入状況について尋ねた結果、早速いくつかの答えが得られ□ました。特に、以下の 4 つの傾向が目立っています。

01 **導入はすでに始まっている:** SLSA と SSDF に組み込まれたソフトウェア サプライ チェーンのセキュリティ対策は、すでにある程度採用されていますが、まだ採用が増える余地が十分にあります。

02 **文化が健全なほど優位に立てる:** 組織文化がソフトウェア開発セキュリティ対策の採用を促す主な要因となっており、信頼性が高く、「過失を責めない」文化がある組織の方が、信頼性が低い組織より SLSA と SSDF の対策を採用する傾向が強くなっています。

03 **重要な統合ポイントがある:** ソフトウェア サプライ チェーンのセキュリティの技術的側面を採用するかは、CI / CD を採用するかに左右されるように思われます。CI / CD は、多くの場合、多くのサプライ チェーン セキュリティ対策の統合プラットフォームを備えています。

04 **予期しない利点がもたらされる:** セキュリティ対策の向上により、セキュリティ リスクの低減のほか、燃え尽き症候群の減少などの他の利点がもたらされます。

セキュリティ脆弱性回避のための今日の企業の取り組み

今年の調査では、各組織が独自に開発したソフトウェアのセキュリティ脆弱性を特定し、解決するために、各組織が現在どのような取り組みを行っているかをより深く理解するため、20 項目以上の質問を追加しました。これらの質問は、大きく分けて以下の 2 つに分類されます。

- 回答者に同意するかしないかを尋ねる質問（たとえば、「私の所属組織は、セキュリティ上の脅威に対処するための効果的な方法を採用している」、「セキュリティテストを実行するために必要なツールを利用できる」など）。

- 回答者に、自分の所属組織でセキュリティ対策がどの程度確立されているか、または確立されていないかを尋ねる質問（たとえば、「ビルドはビルド スクリプトでのみ定義している」、「本番環境のリリースは一元化された CI / CD システムを使用して構築し、絶対に開発者のワークステーションには構築しない」など）。初期のテストでは、回答者がセキュリティ関連の質問に同意する方向に偏っていたため、「確立されている / 確立されていない」の尺度を使用しました。ただし、SSDF に関する質問は、「同意 / 不同意」の回答尺度を使用して、より自然な表現を心がけました。

SLSA フレームワーク（執筆時点では v0.1）では、SLSA の「レベル」に関連する一連のソフトウェア サプライ チェーン整合性対策について説明しており、レベルが高いほど、ソフトウェア サプライ チェーンのセキュリティ保証のレベルも高くなります。私たちは、回答者に SLSA に関連する多くの具体的な手法について質問しました。具体的には、「担当している主なアプリケーションまたはサービスについて、次のような手法がどの程度確立されていますか」と尋ねました。表 1 は、アンケートで取り上げた SLSA 関連の手法に関する文言の一覧です。

現在 v1.1 の **SSDF** は、組織がより脆弱性の少ないソフトウェアを出荷し、残存する脆弱性の潜在的な影響を最小限に抑えるための対策に重点を置いています。SSDF の手法は、SLSA の「レベル」の代わりに、組織の準備、開発中のソフトウェアの保護、安全性の高いソフトウェアの作成、発見された脆弱性への効果的な対応という 4 つのカテゴリに分類されています。アンケートでは、複数の SSDF の手法に関する記述にどの程度同意するか（または同意しないか）を回答者に尋ねています。質問は表 2 にまとめています。

SLSA 手法	アンケートの定義
一元化 CI / CD	本番環境のリリースは一元化された CI / CD システムを使用して構築し、絶対に開発者のワークステーションには構築しない
履歴の保持	改訂履歴と変更履歴が無限に保持される
ビルド スクリプト	ビルドが必ずビルド スクリプトによって完全に定義されている
分離性	ビルドが分離されており、同時実行ビルドや後続ビルドと干渉することがない
ビルド テキスト ファイル	ビルドの定義と構成がバージョン管理システムに保存されているテキスト ファイルに定義されている
パラメータ メタデータ	アーティファクトに関するビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）にすべてのビルド パラメータが含まれている
依存関係のメタデータ	アーティファクトに関するビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）にすべての依存関係が記録されている
メタデータの生成	ビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）がビルドサービス、またはビルドサービスを読み込むビルド メタデータ生成ツールによって生成されている
入力の防止	ビルド実行時、ビルドステップがビルド入力を動的に読み込むことが防止されている（つまり、必要なすべてのソースと依存関係があらかじめ取得されている）
ユーザーによる編集の禁止	ビルドサービスのユーザーがアーティファクトに関するビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）を編集できないようになっている
メタデータの可用性	ビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）がこれを必要とするユーザーに（中央データベースなどを介して）公開され、ユーザーが承諾している形式で送信されている
2 人によるレビュー	改訂履歴が変更されるたびに、提出前に信頼できる 2 人の人によって個別に審査、承認されている
メタデータの署名	アーティファクトがどのように作成されたかに関するビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）が私のビルドサービスによって署名されている

表 1.SLSA 関連のアンケートの質問

注: 回答者には、各質問に対し「まったく確立されていない」、「やや確立されている」、「ある程度確立されている」、「かなり確立されている」、「完全に確立されている」の 5 つの選択肢が提示されました。

SSDF の手法	アンケートの定義
セキュリティのレビュー	私が作業しているアプリケーションのすべての主要機能について、セキュリティ レビューが行われている
コードの継続的な分析とテスト	私たちは、検出されたことのない脆弱性の有無を特定または確認するため、サポート対象のすべてのリリースについて自動または手動のコード分析およびテストを継続的に行っている
早期セキュリティ テスト	私または別のチームがソフトウェア開発プロセスの早期にセキュリティ テストを実施している
脅威への適切な対処	私の所属組織では、セキュリティの脅威に対処するための効果的な方法が準備されている
開発チームへの統合	セキュリティに関する役割が私たちのソフトウェア開発チームに統合されている
文書に関する要件	私たちの組織には、組織が開発または取得したソフトウェア（サードパーティ製やオープンソースのソフトウェアを含む）に対するすべてのセキュリティ要件を特定、文書化するためのプロセスが導入されている。
要件の定期的な見直し	セキュリティ要件が定期的に（年 1 度または必要に応じてより頻繁に）見直されている
メタデータの生成	ビルド メタデータ（依存関係、ビルドプロセス、ビルド環境など）がビルドサービス、またはビルドサービスを読み込むビルド メタデータ生成ツールによって生成されている
開発サイクルとの統合	私の会社では、ソフトウェア セキュリティ プロトコルが開発プロセスにシームレスに組み込まれている
プロジェクト間共通の標準プロセス	私の会社では、プロジェクト間で共通の、ソフトウェア セキュリティ対応のための標準プロセスが導入されている
セキュリティ レポートの監視	私たちは、使用しているソフトウェアとそのサードパーティ製コンポーネントの脆弱性に関する公開ソースの情報を監視するよう、継続的に努めている
必要ツールの準備	私は、セキュリティ テストの実施に必要なツールを利用できる

表 2.SSDF 関連のアンケートの質問

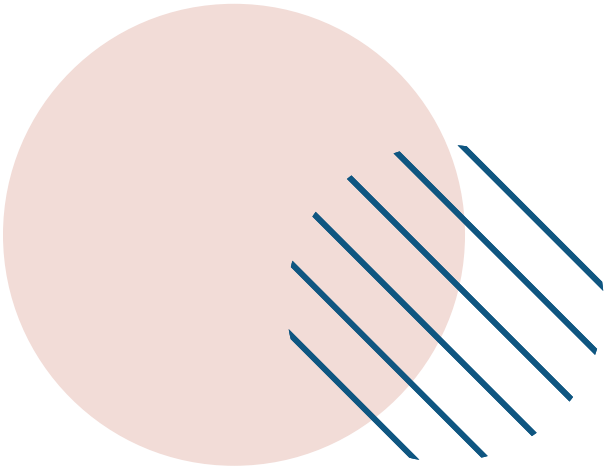
注: 回答者には、質問ごとに「まったくそう思わない」、「そう思わない」、「あまりそう思わない」、「どちらとも言えない」、「ややそう思う」、「そう思う」、「非常にそう思う」の 7 つの選択肢を提示しました。

全体として、業界の新しい手法が比較的広く採用されていることがわかりましたが、これらがさらに確立される余地は十分にあるといえます。たとえば、「私の会社では、ソフトウェア セキュリティ プロトコルが開発プロセスにシームレスに組み込まれている」という記述に 66% が「そう思う」と回答している一方で、「非常にそう思う」という回答は 18% にとどまりました。図 1 と図 2 は、セキュリティに関する質問に対する参加者の回答をまとめたものです。

本番環境のリリースに継続的インテグレーションと継続的デリバリー（CI / CD）のシステムを使用することが最も一般的に確立されている手法であることが確認され、回答者の 63% が「かなり」または「完全に」確立されていると答えました。CI / CD がこのリストの上位を占めた点は、ほとんどの組織が CI / CD プロセスの一環としてアプリケーション レベルのセキュリティ スキャンを実施していることを明らかにした先行のセキュリティ調査とも一致しています。さらに、セキュリティに焦点を当てた別の一連の定性的インタビューから、ほとんどの開発者が、開発中に前述のツールをローカルで実行できないことがわかりました。SLSA フレームワークも同様に、サプライ チェーン セキュリティの中心的統合ポイントである CI システムを基盤としています。次項で説明する私たちのモデル分析により、組織における CI の有無は、その組織のセキュリティ対策の成熟度の予測因子であることがわかりました。

a n organization was a predictor of the maturity of its security practices.したがって、この重要なインフラストラクチャがなければ、組織が自分たちで作成したソフトウェア アーティファクトに対して、一貫した検証ツール、リンター、テストを実行することは非常に困難であると考えられます。

CI / CD のほかに一般的に確立された手法として、コード履歴を無期限保存する（60%）、ビルドの定義は必ずスクリプトで行う（58%）、ビルドは互いに分離する（57%）、ビルド定義はソース管理に保存する（56%）といった手法が挙げられます。一方、最も一般的でない確立された手法は、コード変更の承認に複数人のレビュアーを必要とする（45%）と、改ざんを防止、検出するためにビルド メタデータに署名する（41%）の 2 つでした。



確立された手法に関する質問と並行して、参加者には、所属する組織のセキュリティに関する一連の記述に同意するか否かを尋ねました。最も同意の割合が最も高かったのは、「私たちは、使用しているソフトウェアとそのサードパーティ製コンポーネントの脆弱性に関する公開ソースの情報を監視するよう、継続的に努めている」で、81% の回答者が同意しています。反対に、同意の割合が最も低かったのは、セキュリティ対策がソフトウェア開発に及ぼす悪影響に関する記述で、「自社で採用されているソフトウェア セキュリティ プロセスにより、私が担当するアプリケーションの開発プロセスに遅れが生じている」という記述に同意したのは回答者の 56% でした。

回答者の同意の割合が最も低かったことは心強いことですが、回答者の過半数が現在のセキュリティ プロセスにより開発が遅れていると回答していることは、セキュリティ ツールやセキュリティ対策に改善の余地が大いにあることを示唆しています。私たちのモデル分析もこの解釈を裏付けており、ソフトウェア デリバリー パフォーマンスへの複合的な影響が（わずかではありますが）確認されました。

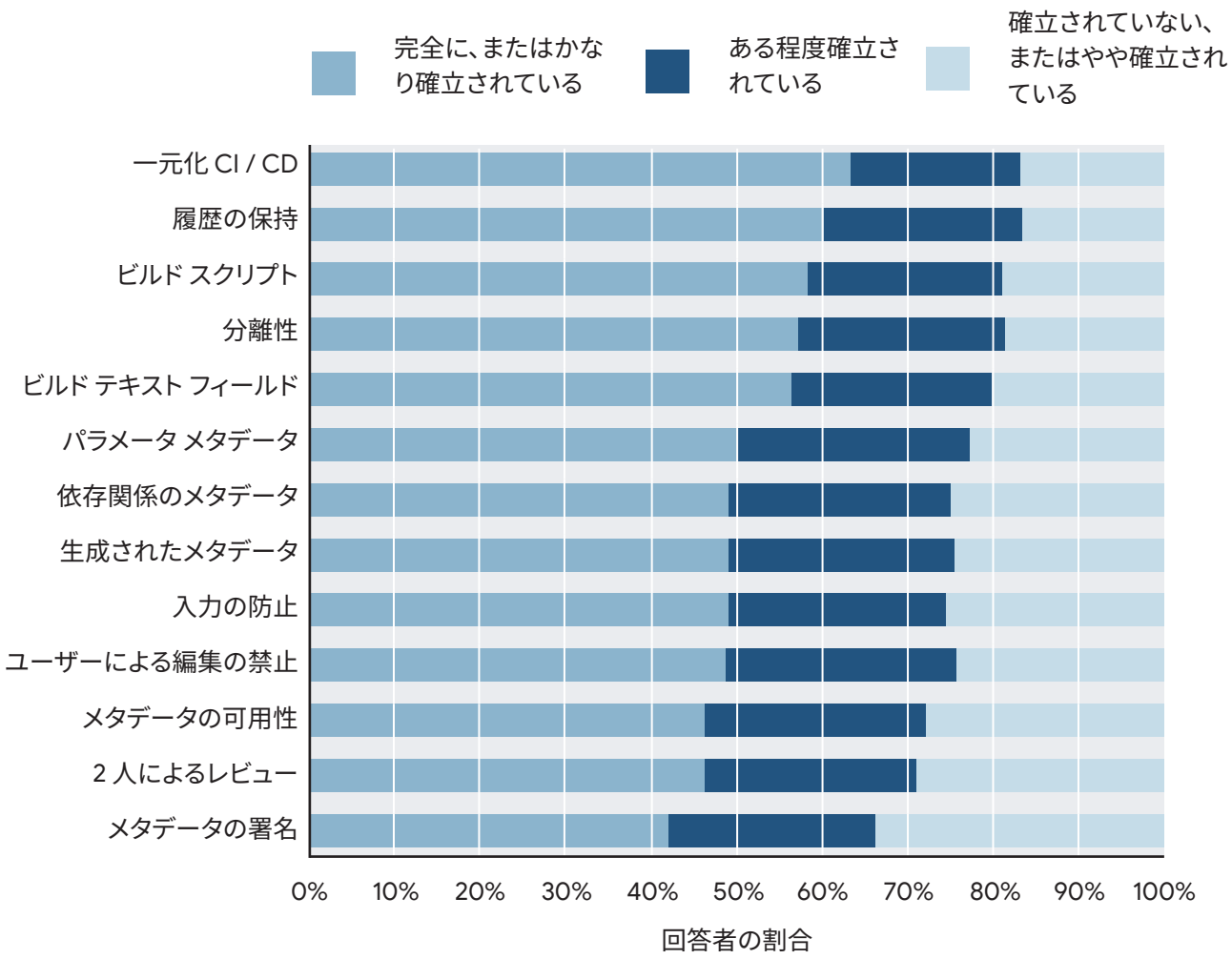


図 1.SLSA 手法の確立

SLSA 手法の確立に関するアンケートの回答大多数の回答者がこれらの手法すべてについてある程度確立されていると回答しましたが、「完全に」確立されていると回答したのは比較的少数でした。

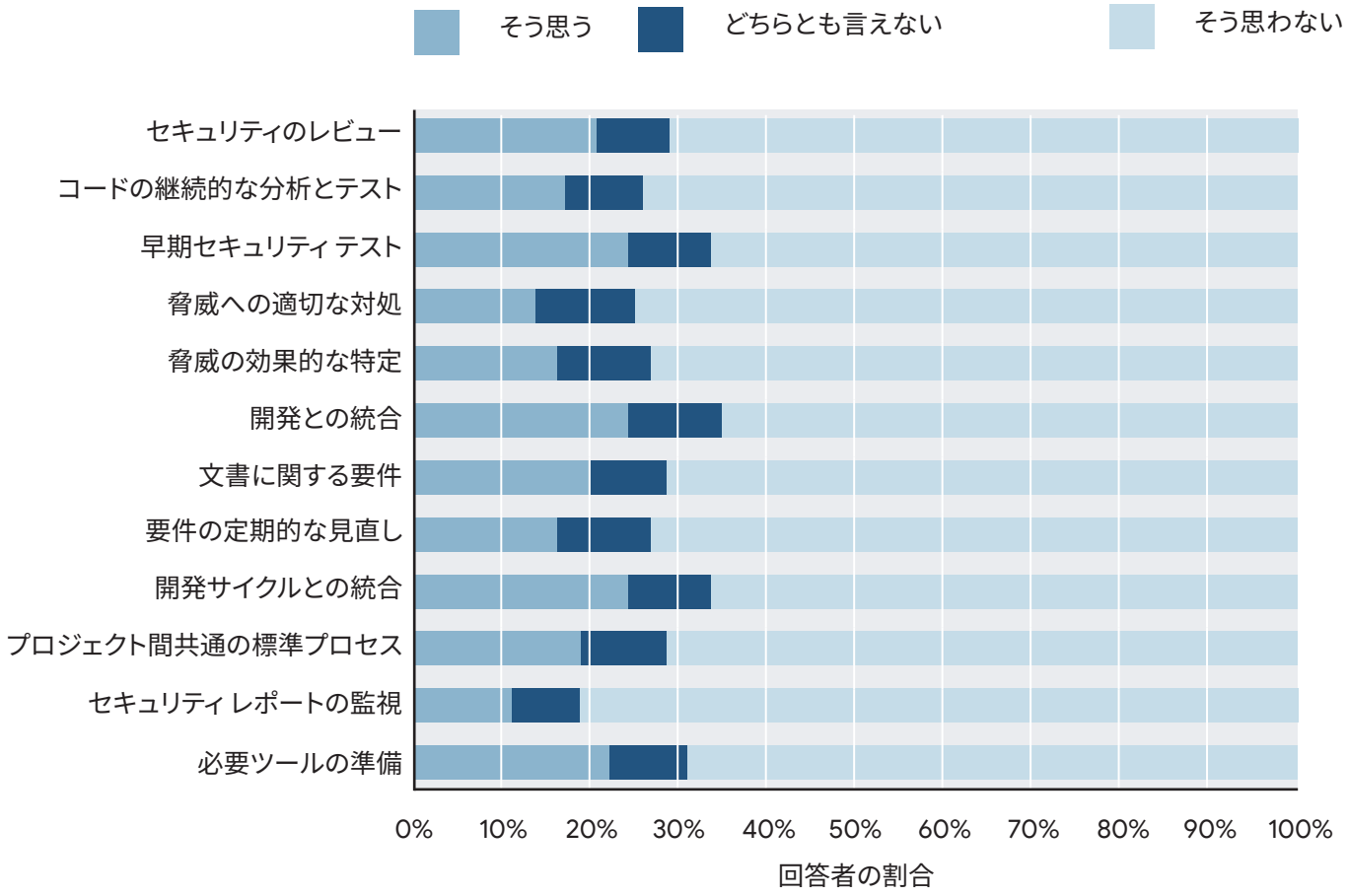


図 2.SSDF 手法の確立

SSDF 手法の確立に関するアンケートの回答SLSA と同様、大多数の回答者が、所属組織がこれらの手法すべてに従っていると回答しました。

会社が適切なセキュリティ対策を導入しやすくなる要因

アプリケーション セキュリティは、ソフトウェア開発の一側面であり、開発者の時間と注意力を要する他の多くの要因の一つにすぎません。大きな摩擦を生むセキュリティ対策は、開発者にとって不満が募るだけでなく、皆が摩擦要因を避けようとするため、全体として効果が得られない場合があります。たとえば、プロのソフトウェア エンジニアを対象とした一連の調査インタビューから、セキュリティ チームとの接点はプロジェクトの開始時か終了時のどちらかに限られており、セキュリティ チームとの連携が困難な場合もあることがわかりました。ある回答者は、「アプリケーション セキュリティ チームはあるが、自分のコードを彼らにレビューしてもらったことはない。ほとんどのエンジニアと同様に、私はたいてい彼らを避けている」と述べていました。

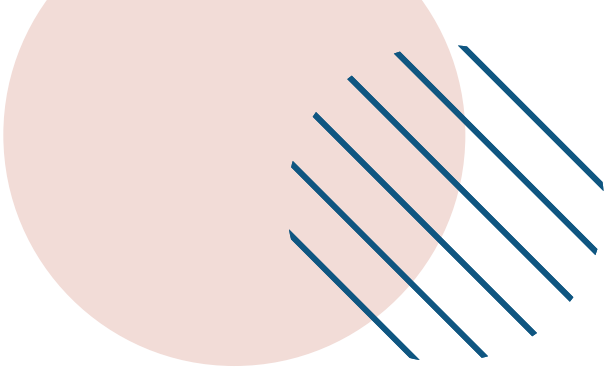
ソフトウェア セキュリティを向上させる手法のひとつとして、セキュリティ対策を妨げる障壁を減らすことが挙げられます。私たちが話を聞いた開発者は、正しいことを行いたいと考えており、多くの場合、起こるかもしれないセキュリティ問題よりも機能や修正の出荷が常に優先されることへの不満を語りました。たとえば、別のセキュリティ関連のアンケートの回答者の一人は、自分たちの最大のセキュリティ上の問題が「セキュリティを優先できないこと」にあり、「セキュリティは、魅力的な分野でもなければ、製品をより多く販売するものでもなく、問題が起こるまで注目されない」と述べました。

この調査データは、開発者が「安全に物事を行いやすくなる」ための複数の要因があることを示唆しています。

私たちが確認した最大の要因は、技術的なものではなく文化的なものでした。**つまり、Westrum の「創造的」文化グループに最も近い組織は、SLSA フレームワークによって定義されたセキュリティ対策が広く確立されていると回答する傾向がより顕著に見られました**[1]。創造的文化の特徴として、高い協調性、リスクと責任の共有、過去の失敗からの学習が挙げられます。私たちは、これらの特徴が、ソフトウェア エンジニアがサプライ チェーンのセキュリティに対してより積極的に取り組むよう促す、職務に関係なくセキュリティに関する取り組みに報いる、潜在的なセキュリティ問題を報告する際に認識されるリスクを減らす、などの多くの形でより健全なセキュリティ手法に表れていると仮説を立てています。

技術的にいうと、セキュリティを高める最も重要な 3 つの要因は、インフラストラクチャに関連しています。これは直感的にわかることです。インフラストラクチャによって脆弱性スキャンや手動コードレビューなどの作業が容易になれば、エンジニアがそれらを活用する可能性が高くなります。特に、**ソース管理、継続的インテグレーション、継続的デリバリーのためのシステム導入**はすべて、SLSA のより確実な実践とも関連があることがわかりました。

[1] 興味深いことに、これと同じ回答者が NIST SSDF の質問に対し「そう思う」と回答した割合は高くありませんでした。SLSA と SSDF は、それぞれアプリケーション開発セキュリティの異なる側面について議論していますが、私たちは、これらの質問群の間に重複が見られると予測しました。前述のとおり、SSDF の回答尺度が「同意」を示す回答に偏っていた可能性があり、それがこの違いの理由とも考えられます。



continuous delivery were all linked with also having more firmly established SLSA practices.このうち重要なのは、セキュリティ問題が開発者の注意を引くタイミングであり、別の調査により、これが主に CI 実施中であることがわかりました。通常、CI はコードレビューの直前の、脆弱性スキャナやその他のコード分析ツールが実行されるときに行われます。これによって、すべてのコード commit が同じセキュリティ要件に従うことが保証されるためです。集中型ビルドシステムがなければ、このような一貫したスキャンがはるかに困難となり、ソース管理が行われなければ、そもそも集中型ビルドシステムを持ちづらくなります。

しかし、CI / CD の一環であるセキュリティ スキャンは、ソフトウェア エンジニアにとって十分早期に行われているとはいえない可能性があります。アプリケーション開発者に対する一連のセキュリティ関連のインタビューでは、開発用ワークステーションでセキュリティ スキャンを実施すれば時間と労力を節約できるとの一貫した回答が得られました。その際、2 つの状況がよく挙げられました。1 つ目は、既知の脆弱性がある依存関係を基にアプリケーションを構築しているかを事前に確認し、その依存関係を基に構築する前にその使用を再検討できるようにしたいということで、2 つ目は、最新の変更によりセキュリティ問題が解決されたかどうかを確認するためだけにときに数時間にも及ぶ長い CI 待ち時間を回避したい、ということでした。どちらの場合も、ソフトウェア エンジニアは、CI の「支え」は必要であるが、同じセキュリティ ツールをローカルで実行できれば、より迅速かつ効率的に作業できる、と述べています。

security tools locally would help them work faster and more efficiently.

上記の文化的要因と技術的要因は、セキュリティの最大の推進要因でしたが、それだけにはとどまりませんでした。その他の注目すべき要因として、以下が確認されました。

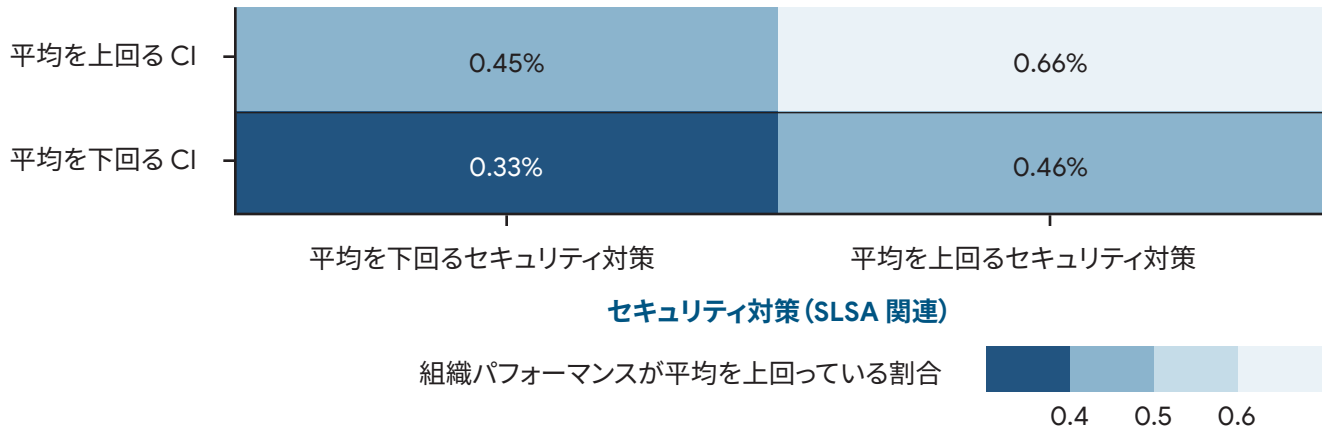
- 勤務形態の柔軟性（たとえば、組織が在宅勤務を認めているか）
- クラウドの利用（パブリック クラウドまたはプライベート クラウド）
- 「クラウドネイティブ」なアプリケーションやサービスでの作業
- 会社がチームを評価し、チームに投資しているという実感
- チーム内の離職率の低さ
- 組織の規模（大規模な組織ほど高いセキュリティ スコアを示す）

しかし、これらの要因は、Westrum の創造的文化（たとえば、柔軟な勤務体系、組織から評価されているという実感、チームの離職率が低いなど）、または CI / CD の採用状況（たとえば、クラウドネイティブ アプリケーションで作業している、または大規模組織で働いている）とおおむね相関しているように思われます。このデータから、組織文化と最新の開発プロセス（継続的インテグレーションなど）が、組織のアプリケーション開発セキュリティの最大の推進要因であり、セキュリティ態勢を強化しようとする組織が最初に取り組むべき要素であると考えられます。

優れたセキュリティ対策がもたらす成果

企業がソフトウェア開発におけるセキュリティ対策を強化することで、どのような効果が期待できるのでしょうか。私たちの調査データから、**企業がサプライ チェーンのセキュリティ対策の確立を進めるにつれ、セキュリティ侵害、サービス停止、パフォーマンス低下の可能性が低くなる**と回答者が予想していることが確認できます。同様に、2022 年前半に実施した別の調査では、CI 実施中に脆弱性スキャナなどのツールを実行することで、ソフトウェアの依存関係の脆弱性を特定する確率が大幅に高まることがわかりました。こういったツールを使用した回答者は、自身のコードまたはその依存関係のいずれかでセキュリティ脆弱性を特定したと報告する確率が約 2 倍高くなっていました。要するに、SLSA と SSDF の手法は、意図どおりに機能しているように思われます。私たちは、これらの手法によりセキュリティ上の脅威を除去できると主張しているわけではありませんが、私たちの調査結果により、これらの手法が組織のセキュリティ リスクを低減していることが示唆されています。

セキュリティ対策は、パフォーマンス ベースの成果にも好影響を与えますが、注意点があります。それは、CI は極めて重要な役割を果たす、ということです。CI が導入されていない場合、セキュリティ対策はソフトウェア デリバリー パフォーマンスに影響を与えません。しかし、CI が導入された場合、セキュリティ対策は、ソフトウェア デリバリー パフォーマンスに大きな好影響を与えます。つまり、基本的には、セキュリティ対策がソフトウェア デリバリー パフォーマンスに好影響を与えるためには、CI が必要ということです。また、一般的に、セキュリティ対策は組織のパフォーマンスに好影響を与えますが、CI が定着しているとこの効果はさらに大きくなります。下のグラフは、この効果を表現したものです。



また、回答者は、確認されたセキュリティ リスクが減少しただけでなく、チームメンバーの**燃え尽き症候群も減少**し、自分の組織を**良い職場**として推薦したくなる気持ちが高まったと回答しています。これらの結果は、ソフトウェア エンジニアにとってセキュリティが「良いのはわかるけれども」という性質のものであることを物語っています。つまり、すでにエンジニアが手いっぱいのところにもうひとつ仕事が増えることになります。脅威が発見されたときに計画外作業やこれを想定した「消火訓練」を行うのではなく、既存の開発ワークフローにセキュリティ対策を組み込むためのツールやプロセスによって、セキュリティ リスクが低減し、開発者にとってやりがいのある仕組みができます。

脅威が発見されたときに計画外作業やこれを想定した「消火訓練」を行うのではなく、既存の開発ワークフローにセキュリティ対策を組み込むためのツールやプロセスによって、セキュリティ リスクが低減し、開発者にとってやりがいのある仕組みができます。

これらを総合すると、**健全でパフォーマンスの高いチームには、優れたセキュリティ対策が広く確立されている傾向もある**ことがわかります（ただし、前述のとおり、引き続き改善の余地はあります）。SLSA フレームワークや SSDF フレームワークのような手法に従うだけでは、文化やパフォーマンスの測定指標をすべて改善できないかもしれませんが、セキュリティのために他の開発優先事項を犠牲にする必要がないことは明らかです。

05

予想外の発見

Derek DeBellis

毎年のレポートでは、その年のアンケートの回答に焦点を当てていますが、私たちは、State of DevOps レポートのカタログ全体や類似の調査（たとえば、燃え尽き症候群や文化に関する調査）の文脈でこれらの調査結果を理解するために最善を尽くしています。再現作業を通じてこれらの効果の信頼性を検証することは、この研究プログラムの中核となる考え方です。これにより、データに適合するように考えを調整し、変化する傾向や新たに出現する傾向を理解できます。

今年、私たちはいくつかの驚くべき事実に突き当たりました。これには、さまざまな理由が考えられます。ひとつは、今年のサンプルには、これまでの報告書よりもキャリアが浅い人が多く含まれていたことです。ひとつの解釈として、技術的手法や機能の導入を監督または指示する責任者ではなく、これらに対し直接責任を負う人々からより多くの意見を聞いたということが挙げられます。

might be responsible for overseeing or directing the implementation of these practices.もうひとつの可能性として、業界や世界で何かが変化したことが挙げられます。昨日までうまくいっていたことが、明日もうまくいくとは限りません。たとえば、マクロ経済の影響や、COVID-19 の影響を受け続けた 1 年の影で、DevOps の物理的特性が変化した可能性があります。最後に、私たちのモデルに含まれるものの微妙な変更により、変数同士の関係が変化した可能性があります。[1]

[1] Judea Pearl の『因果推論の科学「なぜ？」の問いにどう答えるか』と Robert McElreath の『Statistical Rethinking（英語）』は、統計モデルに何を含め、何を含めないかがモデルの出力にどのような影響を与えるかについての素晴らしい例を提示しています。

予想外の発見や仮説に反する発見があると、研究者は、レポートを作成するときに悩みます。その発見が偽りのものであるリスクや、少なくとも複数の研究による実証がまだなされていない（または複数の研究による実証結果と矛盾している）ことを考えると、結果を再現してその原因を理解しようとするフォローアップ研究を行うことが責任ある姿勢となります。[2] 驚きに気を取られることで、研究者は、長年の研究でどれだけの効果が確実に得られたかを過小評価するリスクも犯します。私たちは、「State of DevOps」の各レポートで 100 以上の経路を統計的に調査しています。そうする中で、単に偶然の産物である偽の発見に出くわす危険があります。私たちは毎年、再現性のある研究を行うことで、その対策に努めています。

一方、発見を報告しなければ、お蔵入り効果が生み出されるおそれがあります[3]。お蔵入り効果とは、予期されるもの、または好ましいものがよく知られるようになり、予想外のもの、または好ましくないものは隠されることをいいます。私たちは、バランスを取ることを心がけています。つまり、新しい発見を大げさに取り上げず、これを共有することが重要であるとも考えます。ここでは、私たちが最も驚いたこと、そしてそれが何を意味すると考えているかを紹介します。

01 私たちは、トランクベース開発の手法がソフトウェア デリバリー パフォーマンスに好影響を与えることを一貫して確認しています。実際、これは 2014 年以降の毎年の調査で確認しています。トランクベース開発の機能は、今年、通常とは異なる動きを見せていました。まず、トランクの機能がソフトウェア デリバリー パフォーマンスに悪影響を及ぼしました。これまでの調査レポートとは逆の現象でした。この発見の説明のつかなさから、私たちは、これが今後の調査でも再現されるかどうか確認し、コミュニティにおいてこれを説明できるようなことがあるかどうか、伺いたいと考えています。

02 私たちは、ソフトウェア デリバリー パフォーマンスが組織パフォーマンスに有益であるのは、運用パフォーマンスも高い場合のみであり、多くの回答者の運用パフォーマンスが高くないことを確認しました。これは、ソフトウェア デリバリー パフォーマンスと組織パフォーマンスの関連がもっと明確であった、私たちの以前の調査結果と矛盾しています。

[2] Kerr, N. L（1998 年）。HARKing: Hypothesizing after the results are known（英語）。Personality and social psychology review、2（3）、196〜217。
[3] Rosenthal, R.（1979 年）。The file drawer problem and tolerance for null results（英語）。Psychological Bulletin、86（3）、683。

03 ドキュメント作成手法は、ソフトウェア デリバリー パフォーマンスに悪影響を及ぼしました。これは、これまでの報告とは食い違っています。ひとつの仮説として、特に高パフォーマンスのチームにおいて、ドキュメント作成の自動化が進んでいることが挙げられます。追加データを収集するまでは、この仮説を裏付ける、または反証する証拠はほとんどありません。

04 いくつかの技術的機能（トランクベース開発、疎結合アーキテクチャ、CI、CD など）は、燃え尽き症候群の予測因子と思われました。前述のとおり、今回のサンプルの多くの回答者は、前年のサンプルの回答者よりも著しくキャリアが浅いことがわかりました。したがって、私たちは、計画の策定や監督を担当する人ではなく、機能の実装を担当する人に話を聞いていた可能性があります。実装プロセスは、その監督よりもはるかに困難とも考えられます。この発見をより深く理解するために、さらなる調査を行いたいと思います。

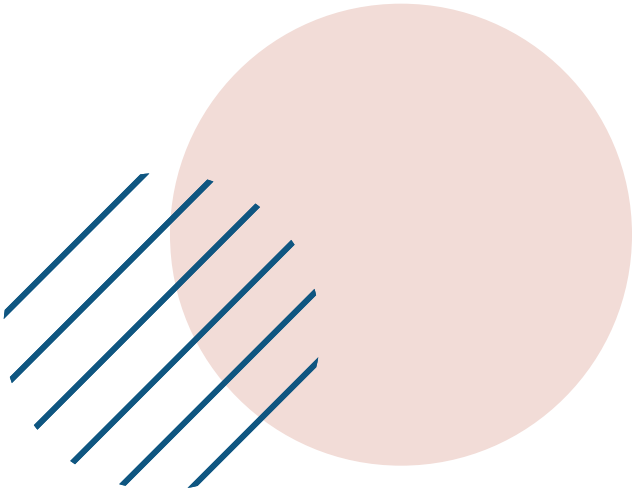
05 信頼性エンジニアリングの手法は、ソフトウェア デリバリー パフォーマンスに悪影響を及ぼしました。ひとつの理由として、これらは必ずしも因果関係にあるわけではないことが挙げられます。今年、私たちは、新しいクラスタリング分析（「他社との比較」の項を参照）で、あるクラスタのサブセットが、信頼性を重視する一方で、ソフトウェア デリバリー パフォーマンスを無視しているように思われることに気づきました。私たちは、一方を実行しなくても他方を実行できるという意味で、これらは切り離されていると考えています。しかし、最終的には、組織パフォーマンスという観点からソフトウェア デリバリー パフォーマンスを重要視するには、信頼性の確保が必要です。

06 安全なソフトウェア サプライ チェーンを維持するために、チームがこれらの主法を採用しているかどうかを把握するため、SLSA 関連の手法を追加しました。セキュリティ対策の実施とパフォーマンス（たとえば、技術的機能の使用、より優れたソフトウェア デリバリー パフォーマンス、より優れた組織パフォーマンス）の間にはなんらかの関連があると予想していましたが、実際には、技術的能力がセキュリティ対策を通じてソフトウェア デリバリー パフォーマンスと組織パフォーマンスに影響を与えることがわかり、驚かされました。

SLSA 関連手法を追加することで、継続的インテグレーション、バージョン管理、継続的デリバリーがソフトウェア デリバリー パフォーマンスと組織パフォーマンスの両方に与える影響の多くを説明できるように思われます。言い換えれば、多くの技術的機能が SLSA 関連手法に好影響を与え、この SLSA 関連手法への好影響を通じて、ソフトウェア デリバリー パフォーマンスと組織パフォーマンスの両方に好影響を与えるという因果連鎖がデータから検出されつつあります。私たちは、この結果を検出するため、媒介分析を行いました。[4] [5] これにより、私たちの SLSA 関連手法の尺度がチームの他の特徴（たとえば全般的なパフォーマンス）を追跡しているかどうか、また、どのような形でセキュリティ対策がソフトウェア デリバリー パフォーマンスと組織パフォーマンスの向上につながるかを調査することになりました。

来年も、これらの効果を再度調査し、前述の新しいパターンを再現して説明できるかどうか、または異常値（これも説明してみる必要があります）として除外する方針を取る必要があるかどうかを確認してみたいと思います。いつものように、皆様からのご意見をお待ちしています。

DORA コミュニティ（http://dora.community）に参加して、今年のレポートで紹介しているこれらの驚きとその他の発見について引き続き考察しましょう。



[4] Jung, Sun Jae.「Introduction to Mediation Analysis and Examples of Its Application to Real-world Data（英語）」Journal of preventive medicine and public health = Yebang Uihakhoe chi vol. 54、3（2021 年）: 166〜172. doi:10.3961/jpmph.21.069

[5] Carrión、Gabriel Cepeda、Christian Nitzl、および José L. Roldán.「Mediation analyses in partial least squares structural equation modeling: Guidelines and empirical examples（英語）」Partial least squares path modeling.Springer, Cham、2017 年。173〜195。

06

ユーザー属性と企業特性

Derek DeBellis

私たちの調査と業界にご協力いただき、ありがとうございます。

調査対象者

8 年の調査により、業界のプロフェッショナルによる 33,000 件以上のアンケート回答を得ている State of DevOps レポートには、チームと組織を最も成功に導くソフトウェア開発と DevOps の手法を掲載しています。今年は、全世界から各種の業界に属する 1,350 人以上の現役の専門家が、より高いパフォーマンスを促進する要素について Google が理解を深められるよう、自らの経験を共有してくれました。私たちの調査と業界にご協力いただき、ありがとうございます。全体として、さまざまなユーザー属性と企業特性にわたる回答者層は引き続き一貫性の高いものです。

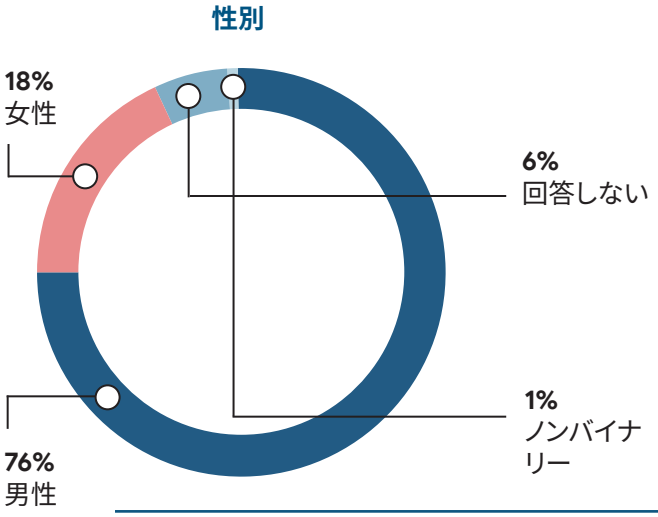
これまでの年と同様に、それぞれの調査回答者について、ユーザー属性の情報を収集しました。カテゴリには、性別グループ、障がいグループ、社会的地位の低いグループがあります。

今年は、会社のサイズ、産業、地域など、企業特性間にわたって、これまでのレポートと一貫した構成比率が見られました。ここでも、回答者の 60% 以上はエンジニアまたはマネージャーとして勤務しており、1/3 はテクノロジー業界に勤務しています。金融サービス、小売、工業 / 製造業の会社からも回答を頂いています。

ユーザー属性

性別

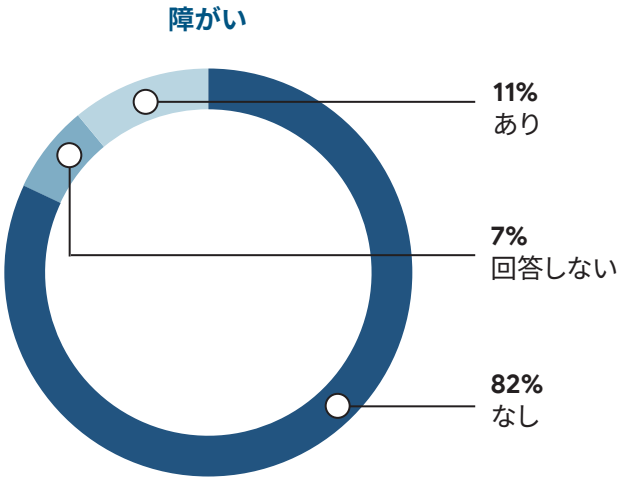
2021 年と比較して、今年のサンプルでは、女性回答者の割合が高くなっていました（12% から 18% に上昇）。男性回答者の割合（76%）は、2021 年（83%）から低下しました。回答者は、チームの 25% が女性だと回答しており、これは 2021 年（25%）と同じです。



女性の割合: **25% 中央値**

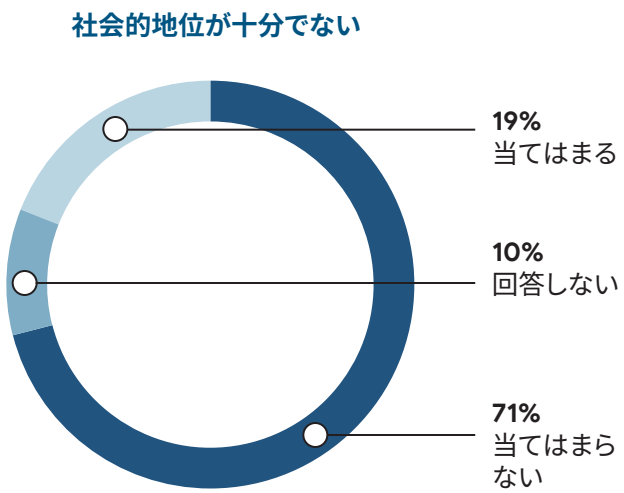
障がい

障がいについては、Washington Group Short Set の基準に従い、6 つの観点で識別しました。障がいについての質問を行うのは、今年で 4 年目です。障がいを持つ人員の割合は、2021 年のレポートと変わらず 11% です。



社会的地位が十分でない

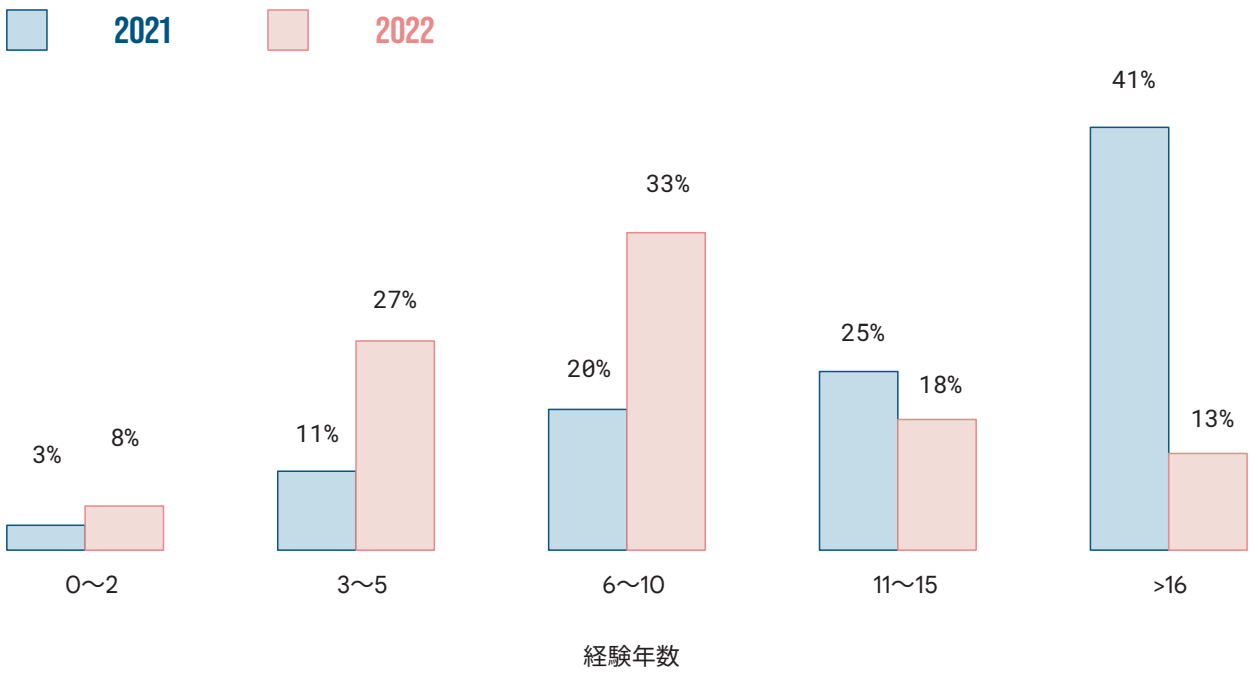
社会的地位の低いグループのメンバーは、人種、性別、その他に関する特性を有しています。社会的地位の低さについての質問を行うのは、今年で 5 年目です。「社会的地位が低い」と識別される人員の割合は 2021 年の 17% から多少増えて、2022 年には 19% でした。



経験年数

経験年数が 5 年以下の回答者は、2021 年 (14%) と比べて今年の方がかなり多く見られました (35%)。おそらく驚くことではありませんが、経験が 16 年を超える回答者の割合は、2021 年 (41%) と比べてかなり下がりました (13%)。

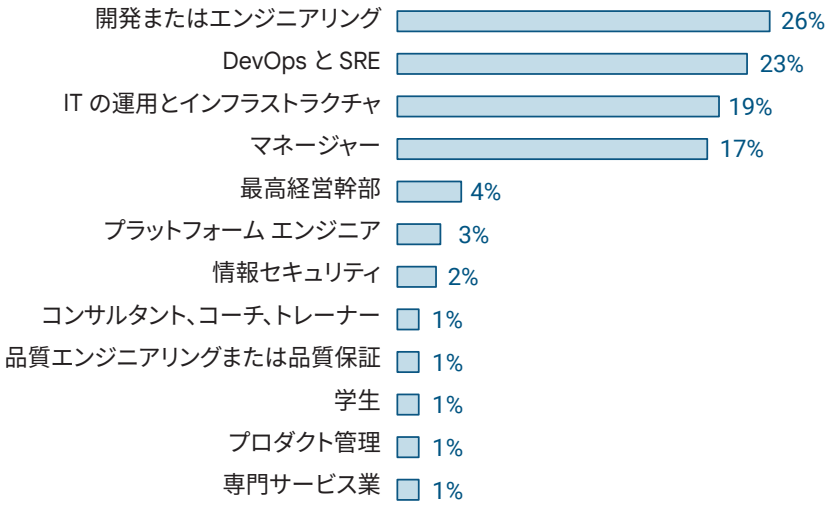
この変動から、データに表れたいくつかのパターンを説明できますが、結果を解釈するとき、特に昨年と比較するときにこの動きを念頭に置いておくことが重要と考えられます。



企業特性

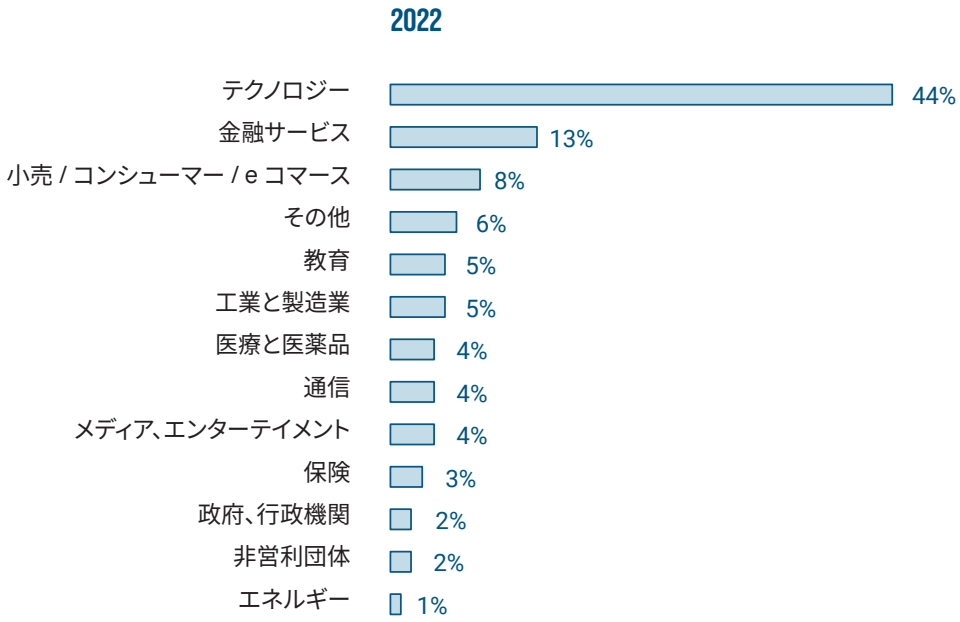
役割

回答者の 85% は、開発またはエンジニアリング チーム (26%)、DevOps チームまたは SRE チーム (23%)、IT 運用チームまたはインフラストラクチャ チーム (19%) に所属する個人、またはマネージャー (17%) で構成されています。IT 運用チームまたはインフラストラクチャ チームに所属する回答者 (19%) の割合は、昨年の割合 (9%) の 2 倍以上となりました。昨年と比較してより顕著に減少したのは、最高経営幹部 (2021 年の 9% から 4%) と専門サービス業 (2021 年の 4% から 1%) です。



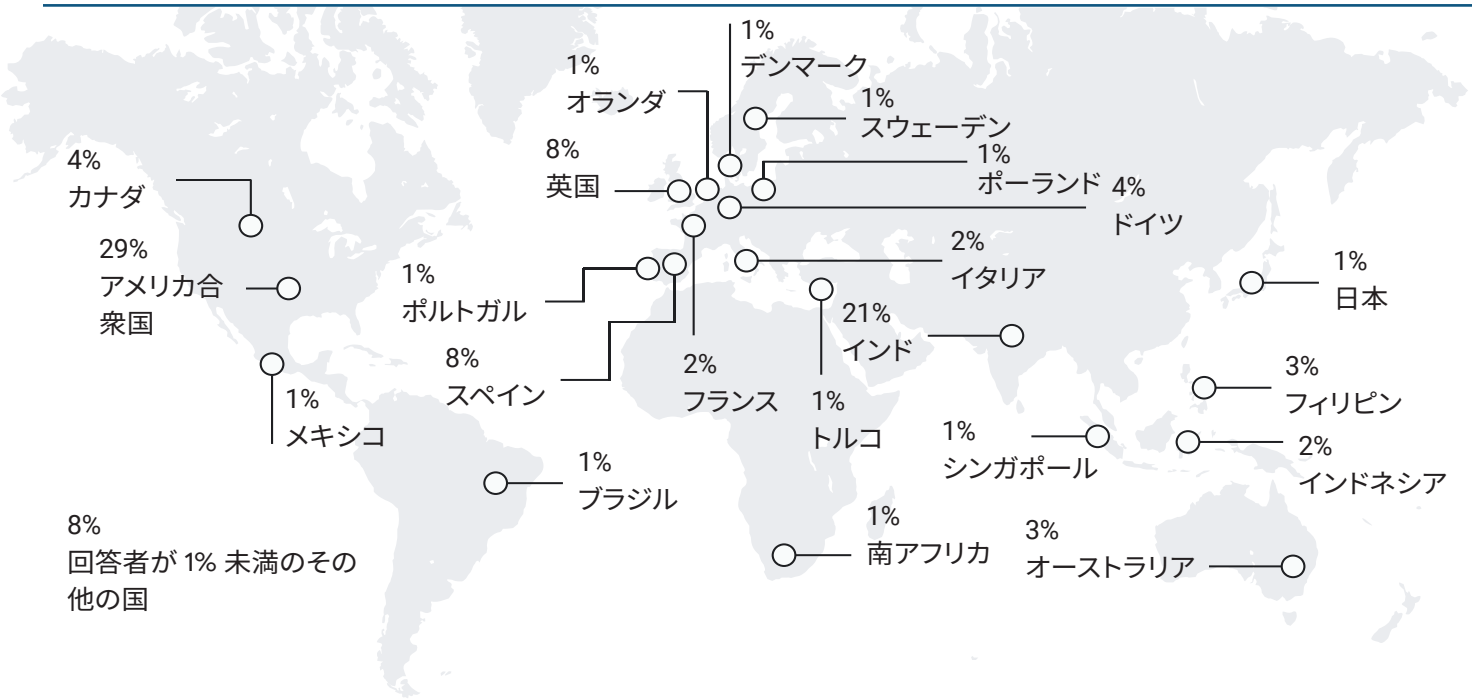
業種

以前の State of DevOps レポートと同様に、テクノロジー業界に従事している回答者が最も多く、その後に金融サービス、その他、小売が続いています。



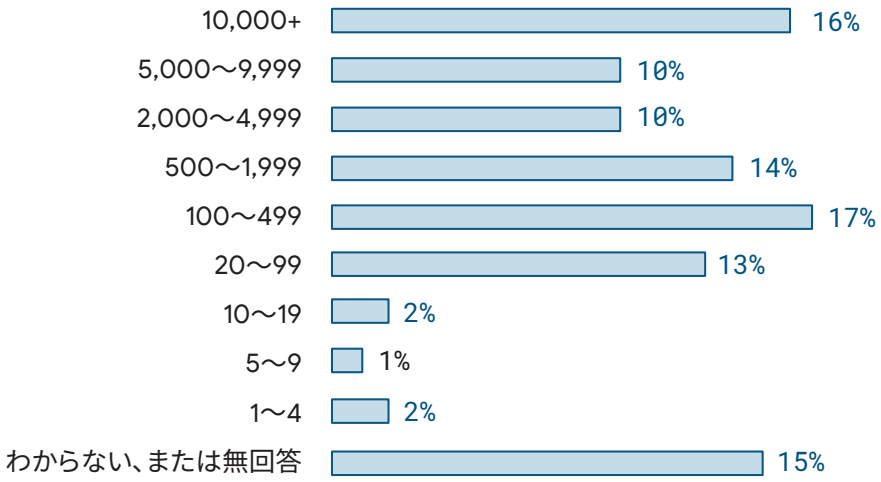
地域

今年は、回答者に対し出身地域ではなく、出身国を選ぶようお願いしました。地域は大陸で表すことが多かったのですが、回答者の構成を把握するには少し粗すぎるようでした。70 か国以上の参加者から回答があり、89% が 22 か国の参加者でした。



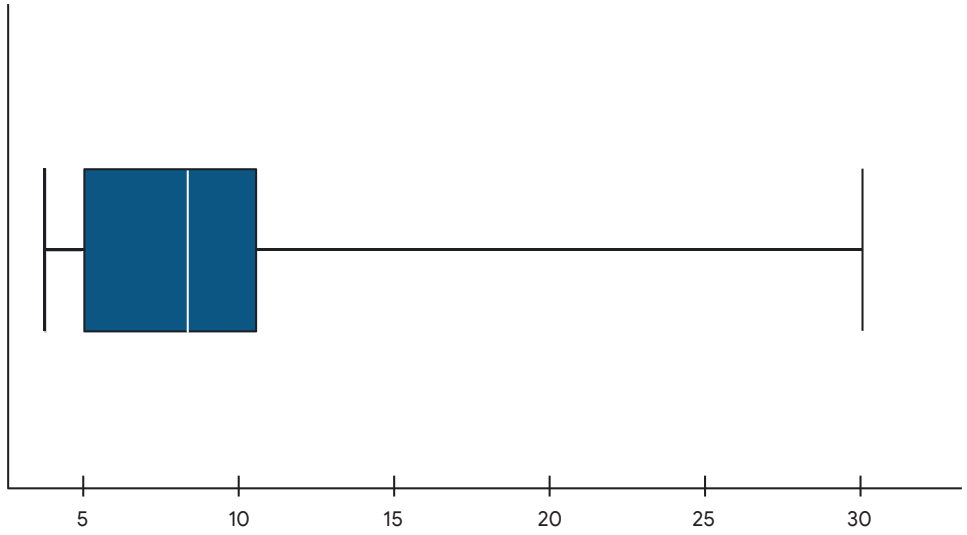
従業員数

以前の State of DevOps アンケートと同様、回答者は、さまざまな規模の組織に所属しています。回答者の 22% は従業員数が 10,000 人を超える企業に勤務しており、7% は従業員数が 5,000〜9,999 人の企業に勤務しています。残り 15% の回答者は、2,000〜4,999 人の組織に所属しています。従業員数 500〜1,999 人の組織の回答者が 13%、100〜499 人の組織が 15%、20〜99 人の組織が 15% と、それぞれにそれなりの割合の回答者がいました。今年は、組織の規模について「わからない」という選択肢も設けましたが、15% の回答者がこれを選択したか、無回答でした。



チームの規模

今年は、回答者に所属チームのだいたいの人数を回答するようお願いしました。25% の回答者が 5 人以下のチームに所属していました。50% の回答者が 8 人以下のチームに所属していました。75% の回答者が 12 人以下のチームに所属していました。

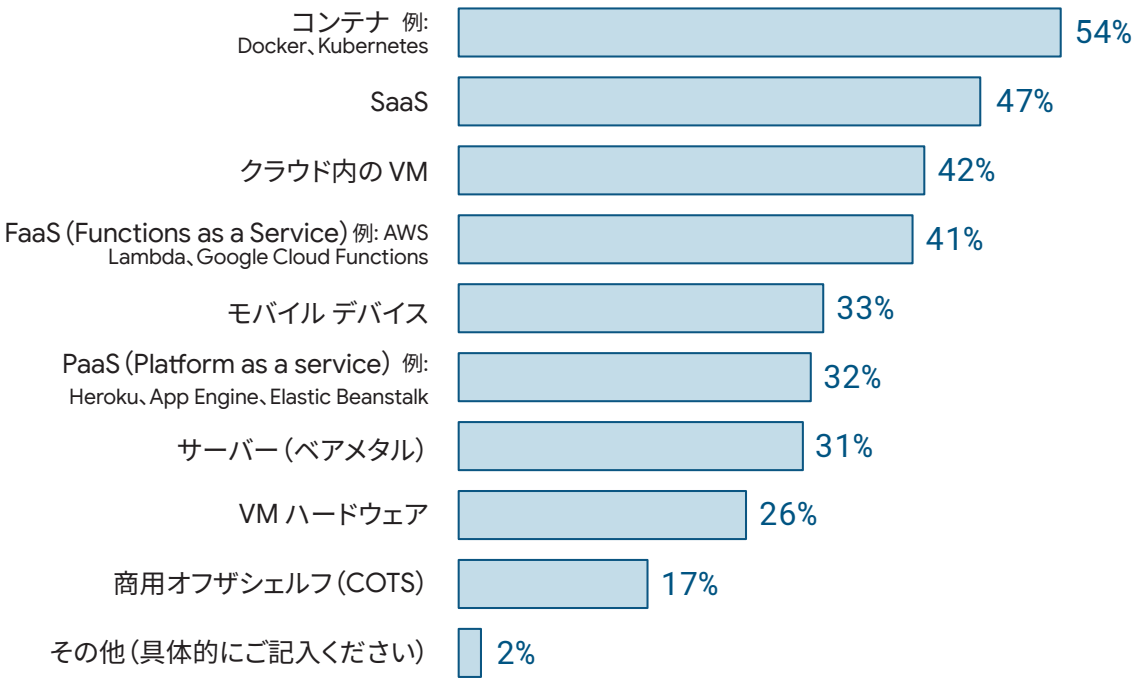


チームの規模

デプロイ ターゲット

2021 年から、回答者が自分たちで扱う主要サービスまたは主要アプリケーションのデプロイ先を確認することにしました。驚くべきことに、最多のデプロイ対象は、コンテナでした。今年も同様でしたが、前年の 64% から 54% に割合が下がっています。

どこにデプロイしているかを回答者がより詳しく回答できるよう、選択肢を追加しました。



07

最後に

Derek DeBellis

このレポートを毎年作成するときは、手法と機能が組織パフォーマンスなどのビジネス クリティカルな成果にどのようにつながるかを厳密に説明するよう努めています。以前のレポートで紹介した多くの効果の再現性を評価し、DevOps の領域で新たに発生した優先事項を考慮して、調査範囲を拡大しました。今年は、セキュリティ対策について深く掘り下げるための調査と分析を企画し、特定の効果の発生条件や依存関係を調べられるよう統計的モデリング手法を変更しました。また、ソフトウェア デリバリーと運用パフォーマンスの状況を説明する新しい方法も模索しました。

多くの点で、今年具体的に示した話は、前年までにも紹介した話です。つまり、技術的な能力は互いに積み重なってより優れたパフォーマンスを生み出すこと、クラウドの利用には多くの利点があること、職場の文化と勤務形態の柔軟性が組織のパフォーマンス向上につながること、従業員が燃え尽き症候群に陥ると、組織が目標を達成できなくなることを示しました。モデルに明示的に追加した相互作用の分析により、特定の効果が発生する条件を理解しやすくなりました。

effects can take place.たとえば、ソフトウェア デリバリー パフォーマンスは、運用パフォーマンス（信頼性）が高い場合にのみ、組織パフォーマンスに好影響を与えるように思われます。これにより、組織として繁栄するためには、両方が必要であるという結論が得られます。また、いくつかの予想外の発見もあり、それについては該当の項で紹介しました。

今年の調査にご協力いただいた皆様に感謝いたします。私たちの調査が、ワークライフ バランスを保ちながら、より良いチーム、より良いソフトウェアを作るための一助となることを願っています。

08

謝辞

多くの熱心な協力者のおかげで、今年のレポートを作成できました。このレポートは、調査の質問設定、分析、著述、編集、レポートのデザインなどの作業の担当者なしには完成できませんでした。著者一同は、こうした作業協力者から今年のレポートのために頂いた協力と教示に感謝します。以下の名前はアルファベット順です。

Scott Aucoin
Alex Barrett
James Brookbank
Kim Castillo
Lolly Chessie
Jenna Dailey
Derek DeBellis
Rob Edwards
Dave Farley
Christopher Grant
Mahshad Haeri
Nathen Harvey
Damith Karunaratne
Todd Kulesza
Amanda Lewis
Ian Lewis
Eric Maxwell
John Speed Meyers 氏
Steve McGhee
Jacinda Mein
Alison Milligan
Pablo Pérez Villanueva
Claire Peters
Connor Poske
Dave Stanke
Dustin Smith
Seth Vargo
Daniella Villalba
Brenna Washington
Kaiyuan “Frank” Xu
Nicola Yap

09

作成者

Claire Peters

Claire Peters は、Google のユーザー エクスペリエンス リサーチャーです。応用環境における DORA のさまざまな側面と表現を調査しています。Google の Cloud アプリケーション モダナイゼーション プログラム（CAMP）、DORA 主導の顧客エンゲージメント、Four Keys 関連のツールを研究しており、チームと個人がより効果的に DORA の基本方針を日々の業務に適用できるよう努めています。Clair は、DORA のコア リサーチチームの一員でもあり、年 1 度の DORA のアンケートと State of DevOps レポートを作成しています。コペンハーゲン大学で応用文化分析の修士号を取得しています。

Dave Farley

Dave Farley は、Continuous Delivery Ltd のマネージング ディレクター兼創設者であり、Continuous Delivery の YouTube チャンネルのクリエイターです。ベストセラーとなった書籍『継続的デリバリー』の共著者でもあります。また、ベストセラーとなった『Modern Software Engineering: Doing What Works to Build Better Software Faster（英語）』の著者でもあります。Reactive Manifesto の共著者であり、オープンソースの LMAX Disruptor プロジェクトでは Duke Award を受賞しています。Dave は、継続的デリバリーのパイオニアであり、CD、DevOps、TDD、ソフトウェア デザインのソートリーダー兼エキスパート プラクティショナーです。長年にわたり高パフォーマンス チーム作り、組織の成功サポート、優れたソフトウェアの作成に携わっています。Dave の詳細については、Twitter、YouTube、ブログ、ウェブサイトをご覧ください。

Daniella Villalba

Daniella Villalba は、DORA プロジェクト専任のユーザー エクスペリエンス調査員です。開発者が快適に作業を行い、生産性を高めるための要素を理解するため注力しています。彼女は Google に入社する前、瞑想トレーニングの利点、大学生の体験に影響を及ぼす心理社会的な要素、目撃記憶、虚偽の自白について研究してきました。Daniella はフロリダ国際大学から、実験心理学で博士号を授与されています。

Dave Stanke

Dave Stanke は Google のデベロッパー リレーション エンジニアで、DevOps と SRE を採用するためのベスト プラクティスについてお客様にアドバイスしています。職歴を通して、スタートアップの CTO、プロダクト マネージャー、カスタマー サポート、ソフトウェア開発者、システム管理者、グラフィック デザイナーなど、あらゆる職務を経験してきました。Dave はコロンビア大学からテクノロジー管理の修士号を授与されています。

Derek DeBellis

Derek DeBellis は、Google の定量ユーザー エクスペリエンス リサーチャーです。Derek は、Google で、アンケート調査、ログ解析、プロダクト開発の中核となるコンセプトの測定方法の特定に注力しています。Derek は最近、人と AI の相互作用、COVID-19 の禁煙に対する影響、NLP エラーを想定した設計、プライバシーの考察における UX の役割に関する記事を公開しました。

Eric Maxwell

Eric Maxwell は、Google の DevOps デジタル トランスフォーメーションの取り組みでリーダーを務めており、そこで、世界のトップ企業に対し、少しずつでも継続的に成長する方法を指南しています。Eric は、キャリアの前半をエンジニアとして過ごし、各所で、さまざまな自動化の作業に関わり、他の実務者の共感を得てきました。Eric は、Google の Cloud アプリケーション モダナイゼーション プログラム（CAMP）の共同創設者で、DORA コアチームのメンバーであり、DevOps エンタープライズ ガイドブックの著者でもあります。Google に入社する前は、Chef Software のエキサイティングな環境で同僚とともに優れたソフトウェアの開発に専念していました。

John Speed Meyers 氏

John Speed Meyers 氏は、ソフトウェア サプライ チェーン セキュリティのスタートアップである Chainguard のセキュリティ データ サイエンティストです。John の調査プロジェクトでは、ソフトウェア サプライ チェーン セキュリティ、オープンソース ソフトウェア セキュリティ、増大する中国の軍事力に対する世界の反応などのトピックが扱われています。John はかつて、In-Q-Tel、RAND Corporation、戦略予算評価センターに勤務していました。John は、Pardee RAND Graduate School で政策分析の博士号を、Princeton's School of Public and International Affairs で広報の修士号（MPA）を、タフツ大学で国際関係学の学士号を取得しています。

Kaiyuan "Frank" Xu

Kaiyuan "Frank" Xu は、Google の定量ユーザー エクスペリエンス リサーチャーです。Kaiyuan は、Google Cloud プロダクトの利用パターンと同プロダクトに関するユーザーの意見を把握するため、ログとアンケート データを分析し、開発者にとってのプロダクトの使いやすさの向上に努めています。Kaiyuan は、Google 入社前、Azure と Power Platform のプロダクトに関する定性的、定量的なユーザー調査を長年 Microsoft で行っていました。彼は、ワシントン大学で人間中心設計およびエンジニアリングの修士号を取得しています。

Nathen Harvey

Nathen Harvey は Google のデベロッパー リレーション エンジニアで、チームがその可能性を実現し、同時にテクノロジーとビジネスの成果を同期できるよう支援する作業を行ってきました。最良のチームやオープンソース コミュニティのいくつかと一緒に作業する機会を得て、それらが DevOps と SRE の原則とプラクティスを適用するための支援を行いました。Nathen は O'Reilly 2020 の「*97 Things Every Cloud Engineer Should Know*」(英語) を共同で編集し、寄稿しました。

Todd Kulesza

Todd Kulesza は、Google のユーザー エクスペリエンス リサーチャーで、今日のソフトウェア エンジニアの働き方、ソフトウェアの将来における働き方の改善方法を研究しています。オレゴン州立大学でコンピュータ サイエンスの博士号を取得しています。

10

手法

調査方法の設計

この調査では、**推論予測**と呼ばれる部門横断的な、理論に基づく設計が採用されています。これは、今日のビジネスと技術の調査で最も一般的に実施されている設計のひとつです。推論予測設計は、純粋に実験的な設計が非現実的または不可能な場合に採用されます。

対象層とサンプリング

この調査における対象層は、テクノロジーと変換の内部、またはその近くで作業を行っている実務担当者とリーダー、特に DevOps に馴染んでいる人々です。調査はメーリングリスト、オンライン プロモーション、オンライン パネル、ソーシャル メディアで、さらには対象の人々に対して、各自のネットワークでアンケートを共有するよう求めること（つまり雪だるま式サンプリング）により、進めました。

潜在的な構成の作成

可能な場合は、以前に検証済みの構成を使用して、仮説と構成を定式化しました。新しい構成は理論、定義、専門家の意見に基づいて作成しました。その後で、意図を明確にし、アンケートによって収集されるデータの妥当性と有効性が高くなるよう、追加手順を行いました。[1]

低パフォーマーと高パフォーマーの差の計算

「他社との比較」の項では、デリバリー パフォーマンスの 4 つの指標について低パフォーマーと高パフォーマーを比較しています。方法は簡単です。デプロイ頻度を例として考えてみましょう。高パフォーマンスのクラスタは、オンデマンド（つまり日に数回）デプロイを行っています。このクラスタが日に平均 4 回デプロイを行うとすると、年間 1,460（4 × 365）回のデプロイを行うこととなります。これに対し、低パフォーマーのデプロイ頻度は 1〜6 か月に 1 回で、平均頻度は 3.5 か月に 1 回、年間約 3.4 回（12 ÷ 3.5）です。高パフォーマーは、低パフォーマーに比べ 417 倍（1,460 ÷ 3.4）デプロイ頻度が多くなっています。この手法は、他の開発パフォーマンス指標にも適用されています。

[1] Churchill Jr, G. A.「A paradigm for developing better measures of marketing constructs（英語）」Journal of Marketing Research 16:1、（1979 年）、64〜73。

統計分析の方法

クラスタ分析

「他社との比較」の項で紹介したいずれのクラスタ分類ソリューションについても、Ward の凝集法 [2] を用いた階層型クラスタ分類を採用して、さまざまなクラスタ ソリューションがデータにどの程度適合しているかを評価しました。

最初に発表したクラスタ分類結果では、デプロイ頻度、リードタイム、サービス復旧時間、および変更時の障害率に関する回答のクラスタを探しました。今年は、クラスタ数を特定するための 30 種類の指標を使用して、14 種類の階層型クラスタ分類ソリューションを評価した結果、3 つのクラスタが見つかりました。[3]

次に提示したクラスタ分析は、方法的には最初のものと同じですが、データ中の複数の項目に対しデプロイされました。スループット（デプロイ頻度とリードタイムの合成）、運用パフォーマンス（信頼性）、安定性（サービス復旧時間と変更時の故障率の合成）に共通の回答パターン（すなわちクラスタ）を見出したいと考えました。また、さまざまなクラスタ分類アルゴリズムを検討し、私たちの手法に対する結果の感度を確認しました。この感度を定量化する方法は（私たちが知る限り）確立されていませんが、見出されたクラスタには類似した特性がある傾向が確認できました。

測定モデル

分析を行う前に、探索的因子分析と、バリマックス回転を使用する主成分分析を使用して構成を特定しました。[4] 抽出された平均分散（AVE）、相関、クロンバックのアルファ [5]、rhoA [6]、異種特性 - 単一特性比 [5][7]、複合信頼性を使用して、収束的妥当性および発散的妥当性と信頼性の統計テストを確認しました。

[2] Murtagh、Fionn、および Pierre Legendre.「Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?（英語）」Journal of classification 31.3（2014 年）: 274〜295。

[3] Charrad M., Ghazzali N., Boiteau V., Niknafs A.（2014 年）。「NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set.（英語）」,『Journal of Statistical Software』, 61(6), 1-36.,「URL http://www.jstatsoft.org/v61/i06/」

[4] Straub, D., Boudreau, M. C.、および Gefen, D.、(2004 年)。「Validation guidelines for IS positivist research」（英語）、Communications of the Association for Information systems、13（1）、24。

[5] Nunnally, J.C.、「Psychometric Theory」（英語）、New York: McGraw-Hill、1978 年。

[6] Hair Jr, Joseph F., et al.「Partial least squares structural equation modeling (PLS-SEM) using R: A workbook（英語）」(2021 年): 197。

[7] Brown, Timothy A.、および Michael T. Moore.「Confirmatory factor analysis（英語）」Handbook of structural equation modeling 361（2012 年）: 379。

構造方程式のモデリング

構造方程式のモデリング（SEM）は、部分的最小二乗（PLS）分析を使用してテストしました。[8] これは、相関に基づく構造方程式モデリングです。

第 2 のクラスタモデルの分析

クラスタのメンバーシップの予測因子を把握するため、多項式ロジスティック回帰を採用しました。[9] この手法を採用した理由は、クラスタのメンバーシップ、今回の場合は 2 階層を超える順不同カテゴリデータを予測したかったためです。クラスタのメンバーシップが予測した結果を理解するため、各結果（燃え尽き症候群、計画外作業、組織パフォーマンス）に対し線形回帰分析を行いました。

[8] Hair Jr、J. F、Hult、G. T. M.、Ringle、C. M.、および Sarstedt, M、（2021 年）。「A primer on partial least squares structural equation modeling (PLS-SEM)」（英語）、Sage publications。

[9] Ripley、Brian、William Venables、および Maintainer Brian Ripley.「Package 'nnet'（英語）」R package version 7.3-12（2016 年）: 700。

11

関連情報

DORA コミュニティに参加して、ソフトウェア デリバリーと運用パフォーマンスの向上について話し合い、学び、協力してください。
http://dora.community

Four Keys 指標の詳細
https://goo.gle/four-keys

DevOps 機能の詳細
https://goo.gle/devops-capabilities

Enterprise ガイドブックで、組織に DORA プラクティスを導入する方法を詳しく確認する
https://goo.gle/enterprise-guidebook

サイト信頼性エンジニアリング（SRE）のリソース
https://sre.google
https://goo.gle/enterprise-roadmap-sre

DevOps のクイック チェック
https://goo.gle/devops-quickcheck

DevOps 研究プログラムを調べる
https://goo.gle/devops-research

Google Cloud DevOps Awards 受賞者の電子書籍を読んで、DORA の手法を導入した他社の事例を学ぶ
https://goo.gle/devops-awards

Google Cloud Application Modernization Program（CAMP）について調べる
https://goo.gle/3daLa9s

データと DORA の指標を活用して、技術的プロセスを変革する
https://goo.gle/3Doh8Km

ホワイトペーパーを読む「DevOps 変革の ROI: モダナイゼーション イニシアチブの効果を数値化する方法」著者: Forsgren, N.、Humble, J.、および Kim, G.（2018 年）。
https://goo.gle/3qECIlh

書籍を読む:『Accelerate: The science behind devops: Building and scaling high performing technology organizations (英語)』IT Revolution。
https://itrevolution.com/book/accelerate

Supply chain Levels for Secure Artifacts (SLSA) フレームワークについて詳しく確認する
https://slsa.dev

Secure Software Development Framework (NIST SSDF) について詳しく確認する
https://goo.gle/3qBXLWk

DevOps 文化: Westrum の組織類型について詳しく確認する
https://goo.gle/3xq7KBV

Open Source Security Foundation について詳しく確認する
https://openssf.org/

in-toto について詳しく確認する
https://in-toto.io/

NTIA.gov のソフトウェア部品表について詳しく確認する
https://www.ntia.gov/SBOM

サイバーセキュリティ: SolarWinds と Microsoft Exchange Incidents に対する米連邦政府の回答
https://www.gao.gov/products/gao-22-104746

CI / CD の一部であるアプリケーション レベルのセキュリティ スキャンについて詳しく確認する
https://go.dev/blog/survey2022-q2-results#security

最後に、以前の State of DevOps レポートをご覧ください。すべて https://goo.gle/dora-sodrsに掲載されています。
2014 年の Accelerate State of DevOps Report
2015 年の Accelerate State of DevOps Report
2016 年の Accelerate State of DevOps Report
2017 年の Accelerate State of DevOps Report
2018 年の Accelerate State of DevOps Report
2019 年の Accelerate State of DevOps Report
2021 年の Accelerate State of DevOps Report